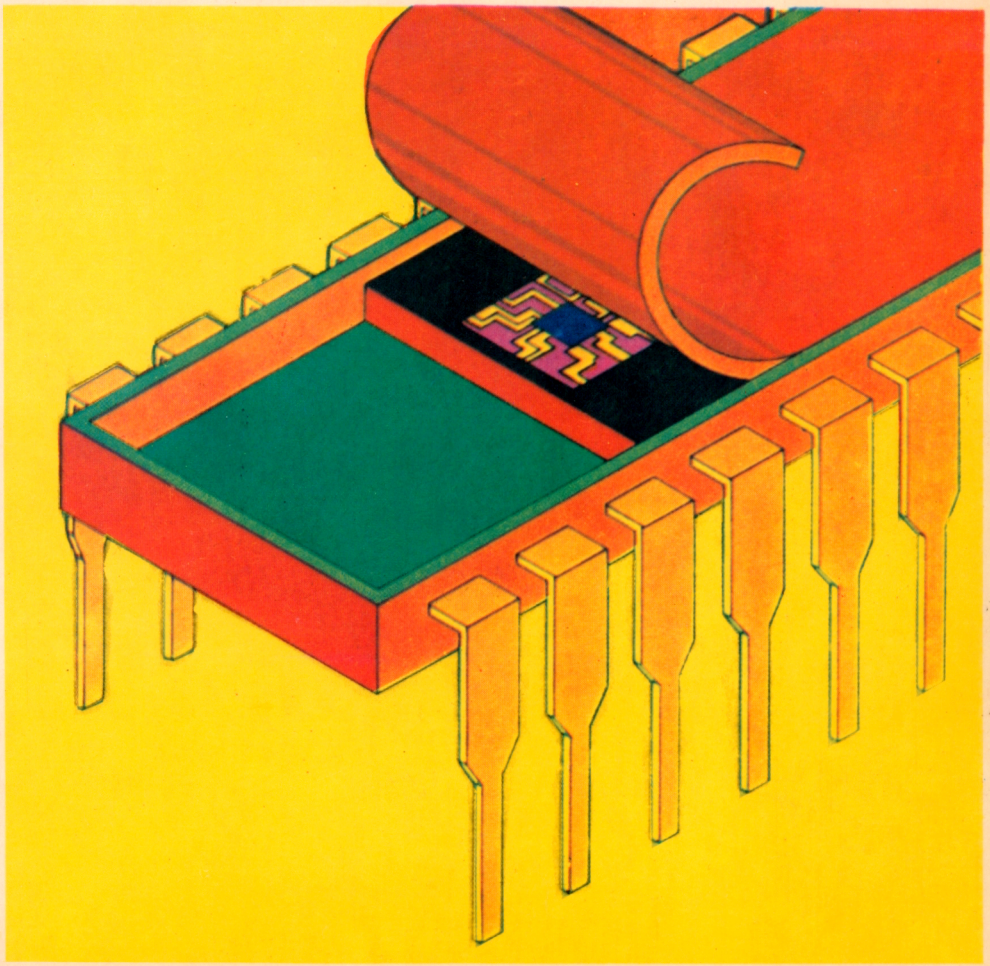
Apple II

APPLE II 監督程式剖視

郭重松 編譯



波前電腦公司出版

APPLE II
監督程式剖視

郭重松 編譯

波前電腦公司出版

Apple II
監督程式剖視

編譯者：郭　重　松
出版者
發行者：波前電腦公司
香港朝光街500號
印刷者：華泰印刷公司
香港仔昌業大厦八樓C座

定價：港幣 $ 26.00

譯　序

當電腦使用者在學會機器的基本操作時，往往會探討機器爲何能如此工作，於是便追踪監督程式的寫法。但往往追踪到某一點看不懂時，便無法繼續追踪下去。本書便是告訴您，到底爲什麼電腦會如此神奇的工作。作者以前呼後應的方式介紹監督程式的內容，包括每一個位元組的定義爲何，實屬難得。

本書必須配合APPLE II REFERENCE MANUAL 後面附錄C（ROM LISTINGS）一起參看，才可以了解。緣此，譯者特將REFERENCE MANUAL 的附錄C，自己加上適當的註解，附錄在本書後面，以方便讀者參看。

郭重松

如何使用本書

監督程式是一部機器的核心，通常爲了要求速度及節省記憶空間，因此其複雜度較一般應用程式高，尤其又是以低級語言編寫，是比較難瞭解的。

讀者在研讀本書時，應對每個常規進入時的狀態，包括暫存器內容，使用欄位內容；執行的功能，包括作了那些動作，設定那些值，叫用或跳至那個常規；離開時的狀態，包括是否返回原叫用 (calling) 程式，暫存器和欄位的內容；等要能徹底瞭解。本書在說明一個功能時，均列出各個有關的常規及使用欄位；並告訴讀者如何在自己的程式中叫系統常規，以達成某些功用。請讀者耐心仔細地研究。

另外有關本書採用的符號也於此略作說明，以免您混淆。本書中每個常規、欄位、暫存器均有個名稱（標誌），以大寫字母表示。如常規名稱COUTI，MON，GETLN 等；欄位名稱爲第零頁的欄位（見第一章）如LOCϕ，WNDLFT，CH等，暫存器是 6502 微處理機的，如A，Y等。欄位或暫存器加括號，表示其內容；如(WND-LET)，(CSWL)，(CH)均是其欄位的內容；如(A)，(Y)表示此暫存器之內容。另外如AIL，H（或PCL,H）是AIL和AIH(PCL和PCH) 之合稱；若提到螢幕某位置，有時以列行來指明，如(CV)，(CH)，其中(CV)是列號，(CH)爲行號。另外十六進位數前有$符號，如$24 爲十進位 36 。凡此細節希望讀者小心體會。

目　錄

前　言 9

導　論 10

概　要 11

第一章
記憶體的分佈 19

19　監督程式所用記憶體的分佈圖

21　　依位址列出的RAM記憶體分佈

22　第零頁

22　　第零頁的欄位

47　第一頁到第三頁

47　　第一頁（ $φ1φφ-$φ1FF ）

47　　第二頁（ $φ2φφ-$φ2FF ）

47　　第三頁（ $φ3φφ-$φ3FF ）

48　　第三頁的位址表

49　第四頁到第七頁和第十一頁

51　　螢幕記憶體位址表

52　週邊控制器工作區域

第二章
輸入和輸出 55

56 鍵盤輸入的分工
57 常規表
64 對鍵盤輸入各常規的叫用
64 鍵盤輸入的叫用表
70 KEYIN常規的取代
72 鍵盤輸入的監督常規
73 位址表1——字元輸入
77 位址表2——列輸入
81 概要——文字輸出到螢幕上
82 輸出到捲動畫面
86 第零頁的欄位
91 捲動畫面的輸出常規
93 由常規控制的螢幕格式
96 經由POKE/STORE控制螢幕的格式
98 捲動畫面的資料處理
99 位址表
102 游標的位置控制
104 位址表
110 一般文字印到螢幕上

111　　位址表
116　　控制字元
117　不用捲動畫面的輸出
121　　位址表
123　　Applesoft 程式範例
124　次顯示區域
124　　將主要抄至次要顯示區
125　爲次顯示頁設定 BASL, H
126　　位址表
128　　直接控制的位址
130　　整數 BASIC 程式範例
131　　Applesoft 程式範例

第三章
插斷處理 133

133 插斷處理
135 NMI 插斷
135 RESET 插斷
136 IRQ / BRK 插斷處理
136 RESET 插斷 —— 舊型的監督程式
137 位址表
140 RESET 插斷 ——自動開始的監督程式
141 設定系統最初的架構
142 冷 / 暖機的決定
142 開電源時的起動
143 系統的重新開始
143 由使用者修飾的RESET 向量
144 位址表
151 IRQ/BRK 插斷
151 IRQ/BRK 插斷的認可
152 IRQ 插斷處理
152 BRK 指令插斷
153 BRK 指令 —— 狀態的保留
154 BRK 指令 —— 舊型的監督程式

154　BRK 指令——自動開始的監督程式
155　位址表

第四章
雜碎整理 159

159 幫助發展機械語言
159 位址表
167 低解析度繪圖
168 第零頁的欄位
169 位址表
172 資料處理功能
173 常規
173 記憶體到記憶體之間的搬移
173 跳至某位址同時作暫存器載入
173 位址欄的遞增
174 保留 6502 暫存器
174 重新取回 6502 暫存器
175 兩個位元組欄位的乘法
175 乘法常規
176 將四個位元組的被除數除以兩個位元組的除數
178 建立RESET向量
179 把十六進位字元轉換爲數值以供使用
179 指令的反組合
180 位址表

184　　Applesoft 資料處理程式範例
185　監督程式的命令處理器
186　　進入監督程式的命令處理器
187　　叫用監督程式的命令處理器
188　　位址表
196　　Applesoft 程式範例
196　經由監督程式使用喇叭
197　　位址表
198　卡式磁帶的輸入和輸出
200　　寫WRITE
200　　讀READ
201　　卡式磁帶輸入／輸出的內部常規
201　　HEADR
202　　RD2BIT
202　　RDBIT
203　　RDBYTE
203　　WRBIT
203　　WRBYTE
204　Paddles, Buttons and Annunciator I/O
205　　GAME I/O的硬體位址表
206　等待常規
207　　WAIT常規的延遲時間
208　控制－Y與參數的使用
209　　Paddle的干擾——程式範例

210 暫存器在BASIC叫用監督程式時的用途
211 十進位到十六進位的轉換
211 Applesoft 程式範例
211 單步和追踪的特色

附錄 213

6502 指令碼
AUTOSTART ROM LISTING

前　言

Apple II 參考手册（ Reference Manual ）中，有一份完整的 Apple II 監督程式，其是以組合語言列出來。Apple II監督程式的剖視（ Apple II Monitors Peeled Manual）（本書）包含監督程式中各個不同常規的說明，和以主題（ topic)來安排的位址表(address table ）說明，而不是以程式在機器中的順序位置來描述。你將發現在這裡所選的和所組合的材料內容，將使得Apple II 的程式設計員，在寫他們的程式時，能夠很方便地使用監督程式裏的常規。

很多在監督程式中的可叫用（ CALL ）點，在多個主題中出現。本書的安排是爲了要減少看書時，頁數翻來翻去地互相參考。所以那些看起來似乎應該在多個主題下說明的地方，將會在每一個應用表中出現。

這份文件涵蓋Apple II的監督程式（包括舊型的監督程式和自動開始（ Autostart ）型式的監督程式），其僅讀記憶體（ ROM ）的位址範圍爲 $F800-$FFFF。這本書並不包括BASIC, APPLESOFT, DOS, HIRES, SWEET16，或者浮點（Floating Point)算術運算的公用常規。

導　論

目前在Apple II 中有兩種可以使用的監督程式 ROM 。這兩種監督程式的功能大部份相同。只有幾個性能不相同。本書描述這兩種監督程式，有些信息僅適用於其中一個時，將會特別有所說明。

有好幾千台Apple II的電腦，以早期型式的監督程式賣出去。在本書中，將被視之爲舊的監督程式。在1979 年，發展出一種新型的Apple II監督程式。這個監督程式含有一些新特性，以利於系統的開動和程式的編輯，但是它犧牲掉了舊型監督程式中的追踪指令的能力（ removing the instruction trace ），單步的特性和十六位元的乘——除常規。在本書中，這個新型的監督程式稱之爲自動開始的監督程式。此自動開始的監督程式可以從Apple電腦公司處和很多電腦經銷商處取得，名爲自動開始ROM， Apple Part No. A2M0027。

要判斷機器是使用那一種監督程式是很容易的。當在電源打開時，機器在螢幕上方有APPLE II的標誌，則此機器是用自動開始的監督程式。如果這台機器是出現監督程式的提醒記號（ * ），則它是舊型式的監督程式。

有一個程式亦可以用來判別出其監督程式是舊型的或者爲自動開始的ROM。在$FAFF（ 64255 或者－1281 ）處的位元組，在自動開始型其值爲 $00 ，在舊型監督程式則爲 $01 。

概　要

第一章

監督程式使用記憶體的分佈（ MONITOR USAGE MEMORY MAP）

監督程式所用的記憶體，和Apple II要控制和顯示至螢幕所需用到的記憶體。

第零頁（ PAGE ZERO ）

詳細描述監督程式在第零頁中所使用的所有記憶體位置，指出值的合理範圍和使用到此位置的所有常規。

第壹頁到第叁頁（ PAGES ONE THROUGH THREE ）

概括性的描述第一頁和第二頁和第三頁中的某些欄位（ fields ）的特別說明。

第四頁到第七頁和第十一頁（ PAGES FOUR THROUGH SEVEN AND ELEVEN ）

說明主文（ text ）如何安排在“螢幕復新記憶體”（ screen refresh memory ）中以便顯示在螢幕上，主顯示區和次顯示區均可作爲文字或低解析（彩色）圖案記憶區。

週邊控制器的工作區域（PERIPHERAL CONTROLLER WORK AREAS）

以圖表來表示在 RAM 記憶體中可用的區域，以便給週邊控制器程式使用。

第二章

鍵盤輸入的分工（KEYBOARD INPUT DIVISION OF LABOR）

描述由監督程式使用的較低層次的常規，以便從鍵盤讀取資料，此包括不讀字只讓遊標移動的副程式。

使用者叫用鍵盤輸入常規（USER CALLS TO KEYBOARD INPUT ROUTINES）

說明使用者如何叫用此程式，以便在所有階層內，皆能從鍵盤輸入文字，並讓使用者的程式可把鍵盤模擬爲一個輸入裝置。

鍵盤輸入的監督常規（KEYBOARD INPUT MONITOR ROUTINE）

表 1 包含經由前節中所描述的常規，由鍵盤逐字輸入有關的位址。表 2 包含從鍵盤作列輸入有關的位址。

概要——文字輸出到螢幕上（OVERVIEW-TEXT OUTPUT TO THE SCREEN）

因爲有許多方式可把文字寫在螢幕上，本節包含下列各頁的蓋觀。在捲頁畫面的文字輸出詳細的描述印資料印在螢幕上的正常方式，

如同使用BASIC中的PRINT，包括在第零頁中的參考表以便執行捲頁畫面的服務。

由常規控制的螢幕格式（SCREEN FORMAT CONTROL BY ROUTINE）

在監督程式內用來控制捲動畫面格式和資料顯示格式的常規之位址表。

經由POKE/STORE控制螢幕的格式（SCREEN FORMAT CONTROL BY POKE/STORE）

描述不用監督程式內的常規而控制螢幕顯示格式的方法。

捲動畫面的資料處理（SCROLL WINDOW DATA MANIPULATIONS）

影響捲動畫面內的資料顯示之常規表，如清除畫面的一部份或把畫面捲動。

游標位置控制（CURSOR POSITION CONTROL）

描述移動游標到相對於現在位置或到一絕對位置的方式。

一般文字印到螢幕上（GENERAL TEXT TO THE SCREEN）

不論其它裝置是否（經由CSWL）建立起來，經由叫用監督程式的常規，放值到A暫存器和叫用COUT常規而把資料印到螢幕上。

不用捲動畫面的文字輸出（ TEXT OUTPUT WITHOUT THE SCROLL WINDUW ）

把螢幕當爲一個已定格式的顯示裝置來處理的方法和意義，其中可以把部分螢幕定爲捲動畫面或者沒有捲動畫面。

次顯示區域（ SECONDARY DISPLAY AREAS ）

將資料放到次文字顯示區域的不同方法。

第三章

插斷處理的概要（ OVERVIEW OF INTERRURT PROCESSING）

插斷的一般和特殊的定義，和在一般電腦中及在特殊的Apple II中的插斷處理。

RESET 插斷 —— 舊型的監督程式

利用位址表且允許使用者叫用到某部份，來說明一個RESET 插斷的處理。

RESET 插斷 —— 自動開始監督程式

利用位址表且允許使用人叫用某部分，來說明一個RESET 插斷的處理。說明軟性入口向量（ Soft Entry Vector ）的建立與使用。

IRQ/BRK 插斷處理（ IRQ/BRK INTERRUPT HANDLING ）

以位址表來描述兩種監督程式對這種型態的插斷之處理。

第四章

幫助發展機器語言（MACHINE LANGUAGE DEVELOPMENT AIDS）

列出監督程式內可被叫用的常規位址表，這些常規可提供檢錯訊息，它把資訊移到記憶體內的某些地方或者經由COUT印出某些信息。

低解析度繪圖（LORES PLOTTING）

描述在監督程式內負責這個功能的常規，列出一個位址表以便能直接叫用它們。

資料處理功能（DATA MANIPULATION FUNCTIONS）

描述在監督程式內負責這功能的常規，它們是把資料由此地搬到另外一個地方，或者改變格式，或在一個常規上運作而能影響到另一個。

監督程式命令處理器（MONITOR COMMAND PROCESSOR）

如何叫用監督程式命令處理器，使它執行監督程式命令且返回到叫用者或者停留在監督程式模式內。

經由監督程式使用喇叭（SPEAKER(BELL) THROUGH THE MONITOR）

此處不談音樂。這是描述如何把喇叭當爲一個發信號裝置來使用，使其工作如同錯誤的鬧鈴或RESET鍵警鈴。

卡式磁帶的輸入和輸出（ CASSETTE TAPE INPUT AND OUTPUT ）

描述所有關於讀或寫磁帶的所有常規，以配合使用者爲了較高層次程式所定的叫用資訊。包括列出每一點的叫用程式。

PADDLES, BUTTONS, AND ANNUNCIATOR I/O

描述機器語言程式設計員如何讀 paddle ，和使用這些裝置所用的位址。

等待常規（ WAIT ROUTINE ）

這個常規將依輸入的A暫存器之值來決定出接掌機器的時間。提出一個時間表和計算式以供使用，在事件之間的間隔很敏感時它是很重要的。

控制Y與參數的使用（ USE OF CONTROL-Y WITH PARAMETERS ）

舉一個機器語言程式的例子，以便迅速地讀取 paddles 。

暫存器在BASIC叫用監督程式時的用途（ REGISTERS FOR BASIC MONITOR CALLS ）

監督程式GO命令程式，使得BASIC得以叫用監督程式的常規，這些常規可由暫存器內接收輸入資料。

十進位到十六進位的轉換（DECIMAL TO HEX CONVERSION）

一個程式範例用來顯示如何從十進位轉換到十六進位。

單步和追踪的特色（STEP AND TRACE PECULIARITIES）

在舊型監督程式內，機器有單步和沒有單步地操作其不同之處。

第一章
記憶體的分佈
MEMORY ALLOCATION

監督程式所用記憶體的分佈圖
MONITOR USAGE MEMORY MAP

記憶體是以256位元組爲單位來區分，通常把它視爲"頁pages"。如同在電腦中大多數可計數的項目一樣，記憶體頁數是由零開始編號。第零頁在這裡面很特殊，其原因爲在第零頁中某一個位元組的完整位址，可以只用一個位元組來表示。很多6502 處理機的指令，僅有兩個位元組的長度（因爲其運算元（operand）是在第零頁中）。因此監督程式對第零頁的使用在下面的章節中特別受重視。

第一頁（位址範圍$φ1φφ-$φ1FF）在Apple II 中亦是非常特殊。這整個256位元組的區域被稱之爲"堆疊器 stack"。此堆疊器對6502 中所提供的一些特殊指令，爲一暫時性的貯存區域（temporary storage area）。A*暫存器或P*暫存器的內容，可以被推放（push）到堆疊器中，其意思是這個暫存器的內容，被貯放在S暫存器目前所指的位置內（即堆疊的最上位置），然後S暫存器的值減1。資料可以從堆疊器中提出或拉出（pulled or poped），此表示S暫存器的值加1，然後S暫存器所指的位元組，則被拿到適當的暫存器

中。 JSR 指令在跳越之前，將會使程式計數器（ program counter ）的內容被壓入堆疊器中。 RTS 指令則從堆疊器中提出兩個位元組，放入程式計數器中。

此監督程式含有某些使用到堆疊器的指令。無論如何，這監督程式在任何時間均不會把堆疊指標暫存器（ stack pointer register ）再定爲某預先值或放一個新值到S暫存器中。

第二頁（位址範圍 $ϕ2ϕϕ-$ϕ2FF ）在Apple II 中是被定義爲鍵盤輸入區域監督程式內。負責讀取鍵盤輸入的常規，將資訊放在第二頁中，以便讓叫用此段常規的程式在下一個“ carriage return ”被發現時，能使用這些資訊。

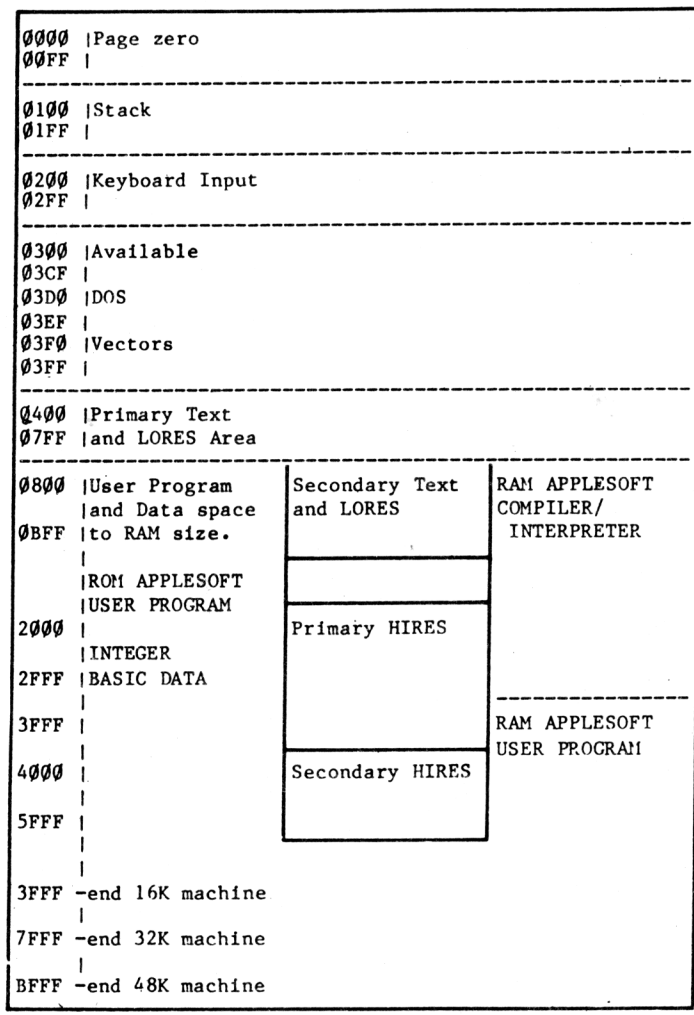
第三頁的定址範圍爲 $ϕ3ϕϕ-$ϕ3FF。這區域的大部分空間沒被監督程式使用。而APPLESOFT或BASIC程式所召用的機械語言程式，時常使用前面 200 左右的位元組。監督程式僅使用最後16個位元組，參考第三頁位址表中所描述的。（注意，無論如何， DOS 使用監督程式所用的16位元組之前面的 32 個位元組。）

第四頁到第七頁包含了主要文字（ primary text ）或彩色圖形(color graphics) 的顯示區域。第八頁到第十一頁包含了次(secondary ）文字彩色圖形的顯示區域。當用到這特性時。一般而言，第八頁常爲使用者區域的第一頁。在位址表中，當以螢幕列來定記憶位址時，第四頁到第七頁和第八頁到第十一頁是一起被說明的。

從位址 $\$\phi 8\phi\phi$ 到機器中記憶體的結尾處，是給使用者的程式和資料用的。但是，如果使用了高解析度的圖形，則記憶體從 $\$2\phi\phi\phi$ 到 $\$3FFF$ 的區域是高解析圖形的主要顯示區域，而位址 $\$4\phi\phi\phi$ 到 $\$5FFF$ 則爲高解析圖的次顯示區。

依位址列出的RAM記憶體分佈

RAM MEMORY ALLOCATION BY ADDRESS



第零頁
PAGE ZERO

監督程式利用第零頁從 32（$2φ）到 73（$49）的位置爲一般功能和正常操作時使用。位置 74-77（$4A-4D）監督程式沒有用到（touched）。位置 78-79（$4E-4F）被修改以提供一隨機數（random number）的開始點以便應用程式使用，如下有說明。

另外，舊型式的監督程式的16位元的乘法和除法常規，使用位置 8φ-85（$5φ-55）（這常規是給解問題的程式使用，監督程式的其它部份都沒使用此區域）。自動開始監督程式並沒有使用這些位置。

自動開始的監督程式在系統起動時，使用位置 0 和 1 。這起動在"RESET Interrupt-Autostart Monitor"重設插斷——自動開始監督程式一節中描述，下面說明位置 0 和 1 的使用。

第零頁的欄位 PAGE ZERO FIELDS

譯註：（x）表示 x 所指位置的內容

十進位位址	十六進位位址	監督程式符號	說明
φφ	$φφ	LOCφ	當電腦電源打開時，自動開始的監
φ1	$φ1	LOC1	督程式在產生自動磁碟靴帶功能（automatic disk bootstrap fun-

十進位位址	十六進位位址	監督程式符號	說明
			ction ）時，使用這些位置。使用這些位置作爲間接定址（ indirect addressing ），而槽的位址（ slot addresses ）被逐一檢查——從槽 7 往下到槽 1 ——以判斷出那一個磁碟控制器存在。如果某一個被發現，則經由 $\phi\phi$-ϕ1 執行一個間接跳越（ Jump Indirect ），以便起動這靴帶動作（ bootstrap operation）。
32	2ϕ	WNDLFT	捲動畫面的左邊行位：（ 譯註：即游標出現的水平位置）範圍是 0 到 39 （ $27 ）。 本欄僅在VTABZ 用到，VTABZ設定BASL,H爲對應於CV 和WNDLFT的記憶體位置。（ 由使用者程式改變）其內容在下一個捲頁動作（ scroll operation ），清除到頁結尾的動作，或 carriage return 輸出時，才開始變成有作用。 CH 存著游標（ cursor ）相對於（

十進位位址	十六進位位址	監督程式符號	說明
			WNDLFT)的水平位置。 改變了WNDLFT 的內容之後，再CALL VTAB或印一個 carriage return 到螢幕時，將使WNDLFT產生效用。
33	$21	WNDWDTH	捲頁畫面（ scroll window)的寬度： 範圍是 1 到 40-(WNDLFT)。 當一個字元經由 COUT寫到螢幕時，它是被放在（BASL），（CH），做完此動作後CH加 1 。然後(CH)與（WNDWDTH)比較以判斷是否游標（ cursor ）已經超過了捲動畫面的右邊邊界（ right margin)了。
34	$22	WNDTOP	捲頁畫面的頂列（ top line ）： 對完全文字螢幕而言，範圍是 0 到 22（$16)。 在圖形與文字混合時，範圍是 2$\phi$ 到 22（ $14 和$15 ）。 在Basics 中VTAB 的有效值爲 21, 22, 23。在一捲頁動作中，這一欄

十進位位址	十六進位位址	監督程式符號	說明
			是用來指出此動作應從那一列開始。它亦是在完成HOME動作後，游標應該擺放於那一列（HOME動作爲將畫面清除。將游標（cursor）置於左上角）。
35	$23	WNDBTM	一般而言，捲頁畫面的底列(bottom line)範圍是（WNDTOP）＋1到24（$18）。 WNDBTM 存著捲動畫面下面第一列的列號。WNDBTM 的內容僅當輸出 carriage return（$80）或 line feed（$8A)時才被測試。它是被清除到"頁的結尾"（clear to End of Page）和捲動常規所使用。
36	$24	CH	距WNDLFT的位移，此處爲下一個送到螢幕的字元要放置的地方：範圍是 ϕ 到（WNDWDTH）－1。在螢幕輸出常規 STOADV將一個字元放到螢幕區域，以當做正常字元輸

十進位位址	十六進位位址	監督程式符號	說明
			出的一部份之後，CH 加1 並與WNDWDTH比較。如果CH 不比WNDWDTH小時，則一個carriage return 將被模擬產生。 注意CH是被GETLN 等常規使用，以把鍵盤輸入在螢幕上顯現。
37	$25	CV	下一個字元應被寫到螢幕上的垂直位置（vertical screen position）（第幾列 line number）：範圍是0 到 23（$17）。CV的內容是相對於螢幕的頂端，而不是捲頁畫面的頂端。這個值可以經由“把希望的列號碼載入A 暫存器且叫用TABV”而設定。也可經由POKE ING把列號存入CV，然後叫用VTAB 而設定。一個字元眞正貯存在螢幕區域是使用BASL,H 以得到列號，而不是使用CV 。上面叫用VTAB或TABV是根據CV（和WNDLFT）來設定BASL,H，以便能馬上供未來參考之用。

十進位位址	十六進位位址	監督程式符號	說明
			如果CV是在WNDBTM 或者位於WNDBTM 下方，在 carriage returns 經過時，CV仍維持在目前的螢幕列上，而捲頁畫面的內容將每次被往上捲動。
38	$26	GBASL	螢幕區域內的記憶體位址。本欄是由GBASCALC 常規設定到適當的記憶體位置，以符合在A暫存器內所指明的列號碼。看位於$2E 的MASK。
39	$27	GBASH	
40	$28	BASL	這兩個位元組的欄位是個記憶體位址，它是在捲頁畫面內目前文字列左端字元的位置。其內容爲CV和WNDLFT 的一個函數。 本欄是由BASCALC常規來設定，以指出“A暫存器內所指的螢幕列的左端字元的記憶體位址”。通常是由VTAB常規來叫用BASCALC，然後VTAB再把（WNDLFT）加到BASL,H，使它們指向捲頁畫面
41	$29	BASH	

十進位位址	十六進位位址	監督程式符號	說明
			列的左端位置。
42	$2A	BAS2L	這兩個位元組僅在一個捲動動作時，才被當做一個工作區域。它是一個目的地列的指標（destination line pointer），在每一列往上移動時使用。
43	$2B	BAS2H	
44	$2C	H2	由HLINE 常規所延長的水平列之右端端點；範圍是 ϕ 到39（$27）。這個位元組是由叫用主程式（calling program）在叫用HLINE之前，先行設定好。
〃	〃	LMNEM	兩個位元組指標（LMNEM, RMNEM）的低位元組，此指標給反組合程式（Disassembler）當做索引（index），指到助憶符號表（mnemonic table）。
〃	〃	RTNL	舊型監督程式的指令追踪常規使用的儲存區。

十進位位址	十六進位位址	監督程式符號	說明
			由VLINE 常規所延長的垂直行之底端：在混合螢幕（mixed screen）而言，其範圍是 ϕ 到 39（$27）。在令螢幕（full screen）圖形，其範圍是 ϕ 到 47（$2F）。這個位元組在VLINE 被叫用前，必須先被設定。注意當清除螢幕（clear screen）（CLRSCR）常規用VL-INE 去清除螢幕時，會用到這個位元組。
〃	〃	RMNEM	與LMNEM 一起被反組合程式（Disassembler）使用，作爲助憶符號表（mnemonic table ）的表索引（table index ）。
〃	〃	RTNH	與RTNL一起被舊型監督程式的指令追踪常規當做一個貯存區域（save area ）使用。
46	$2E	MASK	以此標誌，這個位置由 PLOT使用，根據此點在由 GBASL , H指標所

十進位位址	十六進位位址	監督程式符號	說明
			代表的兩條水平畫線的高部份或低部份，以決定它是 $φF 或 $Fφ 。每個（GBASL），Y 型式的位置在螢幕上含有兩個點，其一點是在一點上面。MASK是用來設定適當的一個，而留下另一個不被改變。
46	$2E	FORMAT	利用這個標誌，反組合程式使用此位元組當做碼的暫時性貯存區域。此碼指出指令的格式，以供顯示時用。
〃	〃	CHKSUM	在讀卡式磁帶時，這個位元組用來連續累積檢查和（checksum），此值將用來與寫入磁帶時所產生的值相比較。這個位元組在開始讀磁帶時，被設定爲零。當每個位元組存入記憶體時，它必須與 CHKSUM 作互斥或的動作（Exclusively ORed ）。在最後一個位元組存入以後，從磁帶中再讀出一個位元組且與 CHKSUM 比較。如果這二值

十進位位址	十六進位位址	監督程式符號	說明
			相等，則確定是個成功的讀取動作。當這個結果最後沒有存回到CHKSUM，則那一欄不能給叫用程式用來決定讀動作是否成功亦或失敗。一種用來作此決定的方法，將於"卡式磁帶的輸入和輸出"部份中討論。
47	$2F	LASTIN	以這個標誌，RDBIT 常規用這個位元組當做一個工作區域，以決定從卡式磁帶輸入暫存器來的輸入（the sense of input）是否已經改變了。
〃	〃	LENGTH	本欄由反組合程式設定，用來表示指令的長度。當反組合指令被輸出以後，PCADJ 利用這個值去計算PCL,H 的新值。此值放在A和Y暫存器內返回到叫用者，以便讓使用者貯存PCL,H 。在舊型監督程式中的指令追踪，亦利用這欄，指出多少個位元組該移入指令追踪

十進位位址	十六進位位址	監督程式符號	說明
			的執行工作區（XQT）。
47	$2F	SIGN	在一個叫用MULPM或DIVPM（在舊型監督系統中，有正負符號的16位元乘法或除法）之後，如果由叫用程式（calling program）將永遠正的結果互補化（complement），則這個位元組的第$φ1位元將被定爲1。
48	$3φ	COLOR	這個位元組存着將放入螢幕的點的色彩碼（在圖形模式）。A暫存器中的較低位四個位元的值定好了再進入SETCOL 常規。這個值然後同時放在COLOR 的高與低半位元組（nibble）。COLOR與MASK同時在設定螢幕區內的位元組的值時被使用，以完成設定一個特殊點爲一選定的色彩。 色彩可以直接設定　經由將彩色碼乘以$11算出的值填入COLOR。舉例而言 color＝orange（9）：

十進位位址	十六進位位址	監督程式符號	說明
			從組合語言　LDA#99, STA color。從BASIC　POKE 48, 9 * 17。
49	$31	MODE	這個位元組由監督程式命令處理常規（Monitor command processing routine）用來控制句子中文法的解析（parsing）和當在十六進位數字之後遇到一空白時的控制動作。舉例而言，一個十六進位位址，其後加一個冒號（colon），將會造成MODE的設定（setting of MODE），所以在未來處理輸入時，每遇到一個空白，則代表一個將被放到記憶體中的十六進位值結束。在解析文句時，MODE的內容表示當命令本身被遇到時，十六進位值應該被貯存在何處，以供使用。MODE根據正號，負號，冒號和句點來設定適當的值。
5φ	$32	INVFLG	這個位元組爲一面罩信號（mask）

十進位位址	十六進位位址	監督程式符號	說明
			，由COUT 1 使用。造成要寫到螢幕區域上的字，是在黑底上顯示白字（INVFLG=$FF）或者在白底上顯示黑字（INVFLG=$3F）或者閃爍（blinking）（INVFLG=$7F）。當RESET 發生時，在SETNORM內的常規會把這欄設爲$FF。此名爲 SETWV 的常規可被叫用以設定相反的視頻（reverse video）。監督程式無法設定閃爍畫面。
51	$33	PROMPT	這個位元組存着提醒字元，這字元被監督程式的GETLN 常規顯示在螢幕上，以表示GETLN 已準備讀取由鍵盤打入的一列字元。當按下RESET 鍵時，舊型監督程式迅速進入 MON 常規，它把 PROMPT 欄設定爲 $AA,"*"。在自動開始的監督程式也是在 MON 常規中，設定"*"爲提醒字元（prompt character），但這並不一定爲處理

十進位位址	十六進位位址	監督程式符號	說明
			RESET 插斷必須的一部份。
52	$34	YSAV	這個位元組是監督程式的命令處理程式（command processor）使用的一個儲存區域。在命令處理機在處理列輸入時，Y暫存器被用作一指標（index）。當一個命令已經被解碼出時，在執行被選定的服務常規之前，Y暫存器是被保留在YSAV內。當回到命令處理程式時，在將控制權轉移到NXTITM，以便繼續掃瞄此輸入列之前，由此欄把Y暫存器的值取回。
53	$35	YSAV1	這個位元組便是Y暫存器的一個儲存區，以避開螢幕輸出常規的召用（怕Y被它毀掉）。Y暫存器是在COUT 1 常規中被儲存和重新取回。
54	$36	CSWL	這兩個位元組的欄位存著“輸出字元的接收和處理常規”的位址。當按下RESET 時，這個欄位被重設
55	$37	CSWH	

十進位位址	十六進位位址	監督程式符號	說明
			以指到COUT 1 ，以把輸出字元送到螢幕上。打入一個監督命令nPc（ n=埠號碼， Pc=control-p ）將使監督程式設定CSWL爲 $\phi\phi$ ，CSWH爲Cn 。本欄所指的常規將接收每個經由COUT "寫入"的位元組（在A暫存器內），命令nPc可視爲一個JMP（CSWL）。 如果執行監督命令" ϕPc "，則CSWL,H被設定爲指到COUT1 ，而不指到C $\phi\phi\phi$ 。
56	$38	KSWL	這兩個位元組欄位存著使用者輸入常規的位址。它是在處理RESET鍵時設定，以指向"取得鍵盤輸入的KEYIN 常規"。監督程式的命令nKc（ n=埠號碼， Kc=control-K ）造成設定KSWL爲 $\phi\phi$ ，KSWH爲Cn 。任何時候監督程式或正在執行的程式經由叫用RDYKE 或叫用其它會叫用RDKEY 的程式，以要求其它位元組的輸入時，這個常
57	$39	KSWH	

十進位位址	十六進位位址	監督程式符號	說　明
			規就被叫用。 監督程式命令"φKc"的執行結果爲將KSWL,H 設定到指向KEYIN，而不是指向Cφφφ。
58	$3A	PCL	這個單元爲程式計數器(Program Counter)的保留和控制區域。除此迷你組合程式(Mini Assembler)用來追踪下一個指令要被放在什麼地方。當一個 BRK 指令被執行，此欄被設定爲指出被65φ2 堆疊存著的位址，此位址爲被執行的BRK指令的下兩個位元組。 這個欄位在監督程式命令L和G(反組合程式和監督程式"GOTO"中被使用到。在指令的反組合時，此欄視需要被增加。在執行監督程式G命令時，此欄被使用爲間接跳越(Jump Indirect)。 每當根據指令的長度而需要增加此欄的值時，透過PCADT 常規的協助來達成。(參考在47或$2F的
59	$3B	PCH	

十進位位址	十六進位位址	監督程式符號	說明
			LENGTH）當從PCADJ返回時，將A放到PCL和Y放到PCH以完成更改動作。 這個欄位由舊型監督程式在監督程式命令S和T（單步指令執行和指令追踪）時使用。針對那些功能，它保存一個指標（pointer），指向下一個要被處理的指令。
6φ	$3C	XQT XQTNZ	這個欄位在舊型監督程式中，是當做指令單步品追踪的一個工作區域。這個單元長度是八個位元組且含蓋A1L,H；A2L,H；A3L,H;和A4L,H。下一個要被執行的指令（由PCL,H的內容指出）被移到此欄內，可能依照指令的型態修改一下，然後在此執行。本欄在自動開始的監督程式中沒有定義。
61-67	$3D-$43		
6φ	$3C	A1L	多用途的監督程式工作區域： 在舊型的監督程式中，可能被指令追踪使用；見前面的XQT。
61	$3D	A1H	

十進位位址	十六進位位址	監督程式符號	說　明
			當監督程式開始處理一個命令時，MODE的起始值被設定爲零。當輸入串被掃瞄時，十六進位數字首先被放入A2L,H 。只要MODE保持爲零，它們又從這裡被移到A1L,H和A3L,H 。當碰到一個正號（plus），負號（minus ），冒號（colon），或句點（period）時，MODE被修改以指出是操作員打的是那種終止符號，而A1L,H 則仍含著那個值。 A1L,H爲BLANK監督程式命令，記憶體檢驗或顯示所用的主要指標（primary index ）。 A1L,H 存著監督程式 ADD 命令的加數（addend）。 A1L,H存著監督程式SUBTRACT命令的被減數（minuend ）。 A1L,H 在監督程式的MOVE命令時，爲被移欄（source field ）的指標（ pointer ）。 A1L,H 是監督程式VERIFY 命

十進位位址	十六進位位址	監督程式符號	說明
			令中所使用的兩個指標(two indices) 其中一個。 如果一個位址已經被指定時，則在L和G監督程式命令和舊型監督程式命令S和T時，A1H,H 爲來源欄，從此處PCL,H 被設定。如果在輸入列中沒有指明位址，則PCL,H便爲上一個更改或使用它的命令所遺留的值。 A1L,H 在卡式磁帶的讀和寫 (READ and WRITE) 監督程式動作時，被使用爲記憶體的指標 (pointer) 。 監督程式的NXTA 1 常規把A1L,H 加1 ，然後此結果與A2L,H 比較。如果A2L,H 小於A1L,H ，則當控制權回到叫用程式時，carry被設定爲 1 。
62	$3E	A2L	多用途的監督程式工作區域： 在舊型的監督程式中，可能指令追踪 ~'obbered ；見前面的XQT。
63	$3F	A2H	

十進位位址	十六進位位址	監督程式符號	說　明
			在監督程式的命令解析時(parsing)，從輸入區域來的十六進位資料存入此欄內。當碰到命令本身時，A2L,H 存著打入的最後一個參數。當MODE含著零時（直到碰到一個正號，負號，冒號，或句號爲止）A2L,H 不斷地被抄到A1L,H和A3L,H。如果碰到一個" 小於 less than " 符號，A2L,H 立即被抄到A4L,H和A5L,H。 A2L,H 被用來終止記憶體檢驗（記憶體顯示memory display)，寫磁帶，讀磁帶，移動記憶體和記憶體測試（verify）動作。 A2L,H 在監督程式SUBTRACT命令的動作中，存著減數(subtrahend)。 A2L,H 在監督程式的 ADD 命令動作中，存著被加數（augend）。 在監督程式的STORE 命令時，A2L,H 爲一來源欄(source field）而A3L,H維持爲指標(po-

十進位位址	十六進位位址	監督程式符號	說明
			inter）。 A2L,H 在一個輸入埠或輸出埠的選擇（控制K或P）命令時，存著埠（port）號碼。 監督程式的NXTA 1 常規把A1L,H加1，然後此結果和A2L,H 比較。如果A2L,H 小於A1L,H，則當控制權回到叫用程式時，carry被設定爲 1 。
64	$4φ	A3L	多用途的監督程式工作區域： 在舊型的監督程式中，可能被指令追踪 clobbered ；見前面的XQT。 如同上面關於A1L,H 的描述一樣，在監督程式命令處理輸入串的掃瞄工作時，A1L,H 和A3L,H 同時被A2L,H填入。 A3L,H 在監督程式STORE 命令處理時，當做一個目的地的指標（pointer）。 A3L,H 被暫存器顯示常規當做一個工作區域使用；此常規被控制E
65	$41	A3H	

十進位位址	十六進位位址	監督程式符號	說　明
			（control-E）監督命令叫用，亦或當做舊型監督程式單一週期或追踪（single cycle or trace）動作的一部份。
66	$42	A4L	多用途地監督程式工作區域：
67	$43	A4H	在舊型的監督程式中，可能被指令追踪 clobbered ；見前面的XQT。 當監督程式的命令處理器（command processor ）掃瞄此輸入區域，而遇到一個"＜"字元時，把A2L，H的內容載入此欄之和A5L，H ）。 在監督程式的MOVE命令執行時，A4L，H 是接收欄的指標（receiving field pointer ）。 在監督程式的驗證（verify ）動作時，A4L，H 是第二欄的指標（second field pointer ）。 監督程式的NXTA4 常規把A4L，H 加1 ，然後往下跳到NXTA 1 ，它把A1L，H 加1 ，然後此結果與A2L，H 比較；如果A2L，H 小於

十進位位址	十六進位位址	監督程式符號	說明
			A1L,H ，則在控制權回到叫用程式時， carry 被設定爲 1 。
68	$44	A5L	多用途地監督程式工作區域：
69	$45	A5H	這個欄位並不在 XQT 區域的範圍內， XQT 在舊型地監督程式，含蓋著A1L到A4L。
注意：			
A5H=$45=Acc			如同上面有關A4L,H 的描述，此欄是由A2L,H 填入。無論如何，在監督程式內是不會被參考到的，除非Acc如亦是A5H。
69	$45	ACC	這五個位元組的欄位爲暫存器的儲存區域。除了下面所指的爲例外， 6502 的暫存器均由SAVE常規貯存起來而由RESTORE常規重新取回（ reloaded ）。 S 暫存器由SAVE貯放在SPNT，但是不再重新取回（ reload ）。 在一個 IRQ 插斷或一個 BRK 指令的執行時，A 暫存器被 IRQ 常
7ϕ	$46	XREG	
71	$47	YREG	
72	$48	STATUS	
73	$49	SPNT	

十進位位址	十六進位位址	監督程式符號	說明
			規存放在 ACC 裡。在一個 BRK 時，進入 SAVE程式的入口點是在 SAV1 標誌，以便貯存其餘的暫存器。其它的暫存器在 IRQ 插斷時，並沒有被監督程式貯存起來。 如同上面所描述地，這些暫存器在執行一個 BRK 指令時被貯放在這個區域。 當執行完一個 BRK 指令或者執行監督程式的命令。控制E時，這個區域的內容是用來把"暫存器"顯示於螢幕上。 執行監督程式的G命令，在跳到所要求的位置之前，這些暫存器（ S 暫存器除外）的值由此區域載入。 在舊型監督程式的單步和追踪（ Step and Trace ）命令常規中，這些暫存器的內容在每個指令執行完後放在這裡，且在下一個被追踪指令要被執行之前，重新取回。
74	$4A		沒有用到

十進位位址	十六進位位址	監督程式符號	說明
75	$4B	沒有用到	
76	$4C	沒有用到	
77	$4D	沒有用到	
78	$4E	RNDL	隨機數（random number）欄位，16位元： 此欄在測試是否按鍵時，由KEYIN常規不斷地往上計數。所以，當它因太長而滿溢（overflow）和重新開始時，其結果仍是有效的隨機值。在監督程式內，沒有其它地方參考到此欄。
79	$4F	RNDH	
8φ	$5φ	ACL	這三個兩一位元組的欄位僅在舊型監督程式中的乘法和除法中用到。在監督程式的任何地方都不會叫用這些常規。因此，僅是在使用者的程式利用到這乘法或除法常規時，才使用這些欄位。 在資料處理功能（Data Manipulation Functions）的單元中時，對乘法和除法常規有個完整的描述。
81	$51	ACH	
82	$52	XTNDL	
83	$53	XTNDH	
84	$54	AUXL	
85	$55	AUXH	

第一頁到第三頁
PAGES ONE THROUGH THREE

第一頁 ($0100-01FF)

第一頁爲一硬體堆疊區域（the hardware stack area）。監督程式僅在6502指令使用到此堆疊（stack），如PHA，JSR，RTS 等，使用到這個區域。此監督程式並不在一RESET 或開電源時插斷或在任何其它時間，設定堆疊器指標的值（S暫存器）。

第二頁 ($0200-02FF)

第二頁爲鍵盤輸入緩衝區域（the keyboard Input buffer area）。在標誌爲"GETLN"處，X暫存器被設定爲一個索引（index）。在標誌爲ADDINP 處，從鍵盤讀進來的字元，放在第二頁中X暫存器爲指之處。這個結果是當回到叫用程式時，從鍵盤讀進來的字已經貯放在記憶體位置$2φφ和往上的地方，最後一個貯存的字，則爲一carriage return，其碼爲$8D。

第三頁 ($0300-03FF)

在第三頁的高位端部分，含有"一些向量"，以便某些插斷的特殊處理。在本頁的低位端到$φ3CF，常被當做機器語言副程式。從$φ3Dφ到$φ3EF是由DOS使用。

第三頁的位址表 PAGE THREE ADDRESS TABLE

Hex	Dec	功 能
$φ3φφ-$φ3EF	768-1φφ7	監督程式沒有使用到這些地方。
$φ3Fφ-$φ3F1	1φφ8-1φφ9	此自動開始的監督程式使用這個位置以當做 BRK 指令的插斷向量（位址）。
$φ3F2-$φ3F3	1φ1φ-1φ11	這是由自動開始監督程式所使用的 RESET 向量（Vector）（位址），在"RESET Interrupt-Autostart Monitor"章節中描述。
$φ3F4	1φ12	開電源指示器：如果"$A5"與$φ3F3 的內容作互斥或（Exclusive OR）的結果等於 $φ3F4 的內容，則此RESET（軟性進入 soft entry）向量被視爲存放。否則的話，一個RESET 插斷將使自動開始監督程式，做開電源時的起動，如果有磁碟可用的話，這包括DOS 的 boot 。
$φ3F5-$φ3F7	1φ13-1φ15	保留給APPLESOFT 使用（"&"向量指令）。

Hex	Dec	功　能
$φ3F8-$φ3FA	1φ16-1φ18	控制Y向量（指令）。
$φ3FB-$φ3FD	1φ19-1φ21	無遮蔽（Non-Maskable）插斷向量（指令）。
$φ3FE-$φ3FF	1φ22-1φ23	IRQ插斷向量（位址）。

第四頁到第七頁和第十一頁
PAGES FOUR THROUGH SEVEN & ELEVEN

位址範圍$φ4φφ到$φ7FF是主要的文字和低解析度圖形顯示區域。亦即，螢幕顯示硬體將貯存在這部份記憶體內地資訊顯示在螢幕上。

位址$φ8φφ 通常爲使用者貯放一般程式和資料的可用記憶區的開始。而$φ8φφ到$φBFF 是次文字和低解析度圖形的顯示區域。以任何值POKEing"-16299"，則此螢幕顯示硬體就被指使，而把次顯示區的內容（而非主顯示區）顯現在螢幕上。POKE"-163φφ,φ"便能跳回主顯示區。

雖然此硬體將從次顯示區域顯示到螢幕上，監督程式並沒有提供此功能。亦即，在監督程式中的BASCALC和GBASCALC常規，僅將輸入到程式中的列（line number）轉換爲在主要地顯示區域內，適當的記憶體位址。如何使用次顯示區域是在"次顯示區域"部份中說明。

連續的螢幕列並不是位於記憶體的連續位置上。在一螢幕列上的字元，貯放在記憶體中的順序是與在螢幕上順序相同，但是這些列是混合成某種形式，以簡化負責顯示到螢幕上的硬體。下面這個表指出對每一列而言，其最左邊的字元在記憶體中的位置（在主顯示和次顯示區）。

在監督程式中的BASCALC 常規，計算出"輸入到此常規的列號（在A 暫存器內）應在的記憶位址"。利用這些字以標明位於輸入列號內的位元位置（ bit position ），下面指出計算的結果：

輸入列號（ A 暫存器 ） $\phi\phi\phi$ABCDE

記憶體位址（ BASH BASL ） $\phi\phi\phi\phi\phi$1 CD EABAB$\phi\phi\phi$

這個能夠利用"模組化modulo"算術來計算，而不用機器語言的AND 和OR 。對列號碼"L"（ ϕ - 23 ），

ADDR = 1ϕ24 + 256 *（（L/2）MOD4）+（128 *（L MOD 2））+ 40 *（（L/8）MOD 4 ）

螢幕記憶體位址表

SCREEN MEMORY ADDRESS TABLE

列	主要顯示區域		次要顯示區域	
	十進位	十六進位	十六進位	十進位
Ø	1Ø24	Ø4ØØ	2Ø48	Ø8ØØ
1	1152	Ø48Ø	2176	Ø88Ø
2	128Ø	Ø5ØØ	23Ø4	Ø9ØØ
3	14Ø8	Ø58Ø	2432	Ø98Ø
4	1536	Ø6ØØ	256Ø	ØAØØ
5	I664	Ø68Ø	2688	ØA8Ø
6	1792	Ø7ØØ	2816	ØBØØ
7	192Ø	Ø78Ø	2944	ØB8Ø
8	1Ø64	Ø428	2Ø88	Ø828
9	1192	Ø4A8	2216	Ø8A8
1Ø	132Ø	Ø528	2344	Ø928
11	1448	Ø5A8	2472	Ø9A8
12	1576	Ø628	26ØØ	ØA28
13	17Ø4	Ø6A8	2728	ØAA8
14	1832	Ø728	2856	ØB28
15	196Ø	Ø7A8	2984	ØBA8
16	11Ø4	Ø45Ø	2128	Ø85Ø
17	1232	Ø4DØ	2256	Ø8DØ
18	136Ø	Ø55Ø	2384	Ø95Ø
19	1488	Ø5DØ	2512	Ø9DØ
2Ø	1616	Ø65Ø	264Ø	ØA5Ø
21	1744	Ø6DØ	2768	ØADØ
22	1872	Ø75Ø	2896	ØB5Ø
23	2ØØØ	Ø7DØ	3Ø24	ØBDØ

此處亦值得注意的是雖然每列有 40 個字， 24 列總共有 960 個位元組，而上面所描述的記憶體區域，每個顯示區域含有 1024 個位元組。其意思是位於第四頁到第七頁的某些位元組並沒有顯示在螢幕上。這些位元組分八組，每組有八個位元組。這空間已經不用或分配給位於槽壹到槽柒的週邊控制器卡片使用。下面的表顯示這個分配。

很容易錯用了這些位置，而造成潛伏地嚴重後果。注意如果一個螢幕的影像在別的地方被產生且被整堆地移到這個區域，則下面所指出的位置將被修改。如果一個程式以監督程式命令 mmm . nnnR ，從

磁帶中讀入，且如果mmm 是小於 $φ4φφ，則下面表中的位元組將由磁帶中載入。如果企圖將螢幕區域的資料保留放在磁碟內且後來又將它BLOAD 到螢幕區域，結果可能會造成混淆不清。此磁碟控制器卡片或可能是其它的週邊裝置界面卡片，有保留一些控制資訊在這些區域內。舉個例，當已經 BSAVE 在磁碟推動機 1 之後，要在磁碟推動器 2 做上面所提的BLOAD，結果將跳回到磁碟推動機 1 。

參考手册（ Reference Manual ）指出使用者必須注意，捲頁螢幕的定義欄（ scroll window definition fields ）WNDLFT 和WND-WDTH，加起來必須不超過 4 φ。如果沒有注意到此警語，破壞了位於下表中的位元組，將有不幸的結果。

週邊控制器工作區域
PERIPHERAL CONTROLLER WORK AREAS

共同（任何／所有）	槽 1	槽 2	槽 3	槽 4	槽 5	槽 6	槽 7
10 進位 16 進位							
1144 Ø478	Ø479	Ø47A	Ø47B	Ø47C	Ø47D	Ø47E	Ø47F
1272 Ø4F8	Ø4F9	Ø4FA	Ø4FB	Ø4FC	Ø4FD	Ø4FE	Ø4FF
1400 Ø578	Ø579	Ø57A	Ø57B	Ø57C	Ø57D	Ø57E	Ø57F
1528 Ø5F8	Ø5F9	Ø5FA	Ø5FB	Ø5FC	Ø5FD	Ø5FE	Ø5FF
1656 Ø678	Ø679	Ø67A	Ø67B	Ø67C	Ø67D	Ø67E	Ø67F
1784 Ø6F8	Ø6F9	Ø6FA	Ø6FB	Ø6FC	Ø6FD	Ø6FE	Ø6FF
1912 Ø778	Ø779	Ø77A	Ø77B	Ø77C	Ø77D	Ø77E	Ø77F
2040* Ø7F8*	Ø7F9	Ø7FA	Ø7FB	Ø7FC	Ø7FD	Ø7FE	Ø7FF

X位置 2φ4φ（ $φ7F8 ）有特殊的意義。每當一個插斷可能發生

和ROM/PROM的擴展結構（expansion scheme）正在使用時，則把$CN載入此位置，其中N爲在工作中（active）的週邊槽號碼。這個是需要的，以便從所用的插斷軟體返回時，可允許適當的週邊卡片恢復工作。

第二章
輸入和輸出
INPUT AND OUTPUT

螢幕的內定動作（ default operation ）爲一捲動的裝置：新資料打入或輸出均在螢幕的底部，且所有上面的資料是一列列往上移，直到最老的信息從螢幕的頂端消失掉。只要在使用者的程式中多加入了一些工作，則亦可能把螢幕當爲一有一定格式的顯示器（ formatted display ）使用。下面爲使用這種型式的效果說明，和遇到某些情況時的處理建議事項。

由使用者程式所產生而要在螢幕上顯示的字，是一次一個字送給監督程式處理。螢幕輸出處理器（ screen output handlers ）核定控制字元與欲顯示的字，然後根據它們所發現地的結果而做對應的動作。舉例而言，當游標（ cursor ）在螢幕的最底列時，輸出一個carriage return 或者 line feed 的字元，此將造成一個捲動的動作發生。如果這個螢幕目前正被當作一有一定格式的畫面使用，而不是一個捲動裝置，則當游標（ cursor ）位於螢幕底列時，則程式應可以很容易地避免輸出一個 carriage return 或者 line feed。

使用者程式要從鍵盤中讀取資料的最簡單方法爲叫用監督程式，此常規在將控制權交回到叫用程式以前，將讀取一列資料（直到一個 carriage return ）。做了這個以後，輸入的資訊總是在記憶體中的

同一位置備用。因此在使用這種型式的叫用和把螢幕當爲一個格式化顯示器之間，將有衝突存在。當此監督程式正在接收鍵盤的輸入，它把字元回映在現今游標（cursor）在螢幕上的位置。當以一 carriage return 來表示輸入的結束時，監督程式把目前游標列上由游標到右限的東西全部清除（在捲動畫面內）。所以，使用者程式必須確定，在要求從鍵盤輸入以前，游標的右邊沒有重要的資料。

也有可能將螢幕區分成捲動區域和非捲動區域。此種方法會產生很多複雜的問題，所以對於格式化的顯示問題其解決辦法是使得此螢幕爲完全捲動（full scroll），且當不希望有捲動服務時，則避免此捲動服務。

使用捲動和非捲動區域其進入點和資格，將在"沒有捲動畫面的文字輸出"部份中討論。

鍵盤輸入的分工
KEYBOARD INPUT DIVISION OF LABOR

負責支援鍵盤輸入的監督程式之常規被設計爲將鍵盤輸入反映到螢幕上（經由 COUT）目前游標（cursor）的位置，且把被打入的字元貯存在鍵盤輸入區域（keyboard input area）（$φ2φφ-$φ2FF）內，以方便叫用程式的使用。執行中的程式在叫用監督程式的鍵盤輸入常規之前，可以將游標（cursor）定在任何位置（在捲動畫面內

）內。當從鍵盤打入一個 carriage return，會使監督程式的鍵盤輸入常規將控制權交回叫用程式，同時字的數目加上 1 放在 X 暫存器和一個 carriage return 放在輸入區域以當做結束標誌。此程式並不需要檢查螢幕的再充電記憶體（ refresh memory ）而判別打入的是什麼。（ 注意：X 暫存器是以零爲開始，所以如果打入 5 個字元，X 暫存器將反映爲 4 ，雖然眞正傳回的值將爲 5 。X 的加 1 是因爲 carriage return 加進去的關係。）

下面所描述的常規包括在位址表中。下面的章節“使用者程式叫用……User Program Calls ……”，說程式的如何設計，以便叫用這些進入點。每個常規的十六進位位址，正十進位位址，和負十進位位址均在下面的括號中寫出。

常規表

常規	說　明
GETLNZ [$FD67] [64871] [−665]	從此點進入的結果是在跳到 GETLN 以便寫出提醒字元（ prompt character ）讀取資料之前，先輸出一個 carriage return （經由 COUT ）。
GETLN [$FD6A] [64874] [−662]	如同在關於“捲動畫面內的文字輸出”一節中所描述的，在這一點進入時，游標（ cursor ）已經被移到適當的位置。

常規	說　明
	GETLN 印出提醒字元（prompt character）且將X暫存器設定一初值，以指出輸入字放到輸入區域的儲存地方。控制權然後交到NXTCHAR。
NXTCHAR [$FD75] [64885] [−651]	這是在字元輸入環路的頂點。RDCHAR 被叫用以取得一個字元放在A暫存器內。在返回時，A暫存器的內容被測試，以明瞭是否出現 control-U（在鍵盤上爲右向的箭頭）；如果眞的發現 control-U，則從螢幕的再充電記憶體（（BASL），Y）中把值取回放入A暫存器，此假設Y暫存器含著與CH相同的值。 如果A暫存器的值是 $Eφ 或者更大，小寫的字經由與 $DF AND起來，轉換成大寫的字。然後這個字從A暫存器存放到輸入區。 如果這個字爲個 carriage return，則CLREOL 被叫用把畫面列上剩餘部分清爲空白，然後一個有條件的跳越，將控制權傳給COUT，所以COUT的 RTS 出口會將控制權交回叫用程式（calling program），而以X暫存器指在 count + 1 這個輸入字元。即是，輸入的字被放在記憶體位置 $φ2φφ 到 $φ2φφ,X，其中 $φ2φφ,X 含著 carriage return。 如果這個字不是一個 carriage return，則控制權被傳遞到NOTCR 常規，以便顯示在輸出裝置上，且解釋看看是否爲影響輸入列的控制字。

常規	說　明
NOTCR [$FD3D] [64829] [-7ϕ7]	這個常規接到控制權時，其有關的字元是放在（IN,X）中。目前設定的INVFLG 被保留在堆疊器中，當INVFLG 被設定爲$FF，則被反映到螢幕上的字，將是黑底白字。然後此字被放在A暫存器中而叫用COUT。 當從COUT回來時，INVFLG 從堆疊器中被取回（restored）。然在在IN,X中的字元被測試，看是否爲兩個特殊的鍵之一：Backspace（左向箭頭）或（列）消除（cancel）（控制-X）。如果爲Backspace，跳至BCKSPC。如果爲Cancel，跳至CANCEL。如果（IN,X）不爲Backspace，亦不爲Cancel，則X暫存器的值被測試，以判斷是否輸入區域滿了亦或幾乎滿了。如果在輸入區域內超過247個字，則叫用Bell，以便通知操作者，這個區域已經快滿了。
NOTCR1 [$FD5F] [64863] [-673]	經過或者沒有此臨界警告的鐘聲（the margin warning bell），控制權便交給了這個常規。在這裡，X暫存器加1，指到輸入區域中下一個要被填入的地方。如果其結果是滿溢（overflow）而變爲零，則跳到CANCEL 以便模擬一個cancel 鍵的打入。在正常的情況，在X暫存器加1後，控制權交回到NXTCHAR，以便繼續字元的輸入和列的建立。

常規	說　明
CANCEL [$FD62] [64866] [- 670]	這個程式經由 COUT 印出一個反斜線（back-slash），以指示操作員所發生的反應。控制權然後送到GETLNZ 以啓動一個新輸入列的打入——原來的列已經不在了。
BCKSPC [$FD71] [64881] [- 655]	當進入到這個常規，則退格（ backspace ）字元已經經由 COUT 被印出去，而其結果爲游標（cursor）的退回。如果目前在 X 暫存器的值爲零，則控制權被交回到 GETLNZ 以便印出提醒字元，且重新設定新值以便列的輸入。否則的話，X 暫存器減 1，而控制權交到 NXTCHAR 以便繼續字元的輸入。
RDCHAR [$FD35] [64821] [- 715]	這個常規叫用 RDKEY ，以得到下一個置於 A 暫存器內的字元。在返回後，如果發現 Escape 鍵已經被按下去，則這個程式叫用適當的常規，以便讀取下一個字元且執行所要求的 Escape 鍵功能。在舊型的監督程式中，爲了達到此目的，將控制權交給 ESC 1 常規，再 JSR 到 RDKEY 以讀取下一個字元。在自動開始的監督程式中，當在 RDCHAR 測試到一個 Escape 字，則將控制權傳給 ESCNEW（經由 ESC 和 RDKEY），它能處理按下一個 Escape 鍵之後再有多個 escape 鍵的功能。 在任何 escape 功能已經被執完後，控制權將回到

常規	說　明
	RDCHAR，如同沒有發生過插斷一樣。
RDKEY [$FDφC] [64784] [-756]	這個常規將位在螢幕復新記憶體區域的BASL,H,CH（把CH的值放入Y暫存器）位置內的字元取出並保留在A暫存器內。然後它將記憶體中的這個字元改爲對應的閃爍字元，以指出目前游標（cursor）的位置。 這個程式要求下一個輸入的字放在A暫存器，此是經由KSWL,H做一個間接地跳越（indirect jump），KSWL,H普通是指向KEYIN。因此返回是回到RDKEY的叫用者，而不是返回到RDKEY本身。
KEYIN [$FD1B] [64795] [-741]	這是一個從鍵盤硬體處取得下一個輸入鍵的常規。這個常規作了兩個必須的動作和兩個額外的動作。這必須的動作爲反復地讀取鍵盤輸入緩衝器，直到它能決定出（經由$8φ位元的出現）在正需要讀一個字時爲止。在這個情況，鍵盤輸入緩衝器視爲在φ2φφ處的$100 位元組緩衝器，而不是$Cφφφ 位置。在檢驗位於$Cφφφ 的狀態，以判別正負旗號是否定爲1。如果那個值爲正值，則此常規廻返KEYIN。如果那個值爲負值，則在$Cφφφ 的值被取且參考鍵盤strobe信號以準備下一個鍵盤輸入。 這個常規所作的輔助動作，第一爲計數隨機數欄（ra-

常規	說　明
	ndom number field），（不管滿溢否）。而第二爲將由RDKEY 常規所修飾過的字元，重新放回(restore)到螢幕區域，以便除掉這閃爍現象。這個重新放回動作是將A暫存器貯存到（BASL），Y，此動作是假設RDKEY 已經將值填入到A暫存器了。在鍵盤暫存器被讀入到A暫存器內之前，已經完成了這個動作。 回到RDKEY 的叫用者（caller）是經由一個RTS完成的。
ESC [$FD2F] [64815] [- 721]	如果發現A暫存器是含著Escape鍵的碼，則這段常規是由RDCHAR進入。此常規經由叫用RDKEY ，而再度把新鍵碼放入A暫存器。在舊型的監督程式中，它然後叫用ESC1 以執行其所要求的單一功能(single function ）。在自動開始的監督程式中，則叫用ESCNEW以執行所要求的功能。在任一個情況中，均利用ESC常規的地位，以便終止Escape 鍵處理的RTS能將控制權交回到RDCHAR 。
ESCNEW [$FBA5] [64421] [- 1115]	這段常規僅在自動開始的監督程式中存在。它是一段負責游標（cursor）移動而不作資料傳遞的常規；此Escape 鍵對Ⅰ，Ｊ，Ｋ，和Ｍ有作用。如果下一個按下去的鍵是這四個的其中一個，則適當的“oed ”功能

常規	說　明
	（Escape 對C,B,A,D 的功能）被叫用。當返回到ESCNEW時，RDKEY再度地被叫用，以便從鍵盤上取得下一個字元（且依它處理）。 如果按下的鍵不是I,J,K,或M，則利用 JMP（而不是JSR）進入ESC 1，因此RTS 將回到ESCNEW的叫用者而不是回到ESCNEW本身。
ESC1 [$FC2C] [64556] [- 980]	在舊型的監督程式中，如果RDCHAR 發現A暫存器內是Escape 鍵碼，則會叫用ESC 1常規。在自動開始的監督程式中，在發現Escape 的時候，控制權交到ESCNEW 常規，再由ESCNEW 根據下一個按下的鍵而叫用ESC 1 或者跳到ESC 1。 ESCNEW在叫用ESC 1 之前，將I,J,K, 或M分別轉換成C,B,A或D的相對功能，ESC 1 將返回到ESCNEW。 如果這個鍵並不是I,J,K, 或M，則ESCNEW 常規跳越（JMP）到ESC 1 且設定 carry旗號爲 1，以便執行適當的功能。在這個情況中，下一個 RTS 將控制權交回到RDCHAR 程式。 當ESC1 被叫用，則A暫存器的內容（和 carry 被設定這條件）指出要被發生的動作。控制權被轉移（有條件地跳越 conditional branch）到適當地捲動畫面服務常規（scroll window service routine）以移動

常規	說　明
	游標（cursor）而不傳遞資料，或者是清除全部或者部份螢幕，或者是上面的某種組合。

對鍵盤輸入各常規的叫用
CALLS TO KEYBOARD INPUT ROUTINES

下面這一章節中說明如何叫用監督程式中不同的進入點，以組合成鍵盤輸入功能，和將會產生什麼樣的結果。

鍵盤輸入的叫用表

常規	如何組合的說明（Description of Set-Up）
GETLNZ	寫出 carriage return 和提醒字元（prompt character），然後讀一列。 組合： X 暫存器，Y 暫存器，和 A 暫存器是無關的。 CH 是無關的。 C V 應該指向在捲動畫面中，開始輸入的那一列。 BASL,H 是無關的。 結果： CR 被寫出去，如果需要的話將發生捲動現象。

提醒字元經由COUT寫出去。

從鍵盤一個字一個字讀入。每一個字被放在$φ2φφ，X然後X值加1。

每一個字被"回映"到螢幕上游標所指的位置，然後游標再向前一步。

當讀到一個carriage return時，控制權將交回到叫用它的程式。

在返回時：

A暫存器含著一個carriage return碼（$8D）。

X暫存器含著在carriage return之前所讀的字元個數。

Y暫存器含著WNDWDTH的內容。

$φ2φφ,X位置含著一個carriage return。

CH含著零值。

CV含著目前列編號的值。

BASL,H含著CV,WNDLFT的記憶體位址。

畫面上在回映輸入（echoed input）的右端之後此列是空白的。

GETLN　寫出提醒字元（prompt character），然後讀一列。

組合：

X暫存器，Y暫存器和A暫存器是無關的。

CV和BASL,H應該一樣的，指向捲動畫面中某位

置。

CH指出在那列上要放置提醒字元的位置，以便讓此反映鍵的輸入（echoed key input）有所遵循。輸入要在那裡開始的列位址（line address），必須包含在BASL,H內。在打入一個carriage return之後，在CV內的列編號將被計算出來且存放在BASL,H中。

結果：

與上面的GETLNZ 相同，僅註解處例外。

返回時：

與上面的GETLNZ相同。

NXTCHAR 於此可打入字元，但省略"把提醒字元印在螢幕上"的動作。

組合：

X暫存器應該爲零，以便開始貯存輸入的位置是在$\phi 2\phi\phi$。

Y暫存器和A暫存器是無關的。

CV和BASL,H應該是互容的（compatible），此是指向畫面上的某列。

CH指出鍵盤輸入的回映將由何處開始。

結果：

與上面GETLN相同。

返回：

與上面的GETLNZ 相同。

註解：對上述所有情況，均假定Escape 鍵的功能如您所用監督程式的參考資料中之描述；而且有 control-U（向右的箭頭）的功能。當那個字在鍵盤緩衝器內被認知時，它由在目前游標（cursor）處的螢幕記憶體內容取代掉，而放在A暫存器內。

RDCHAR　經由KSWL讀取一個字元：放在A暫存器中回到叫用者（caller）。

組合：

X暫存器是無關的，且將不會被破壞（clobbered）。

Y暫存器是無關的。

A暫存器是無關的。

CV和BASL,H應該是互容的，在捲動畫面中它們是指向輸入要被放置的列位置。CH指出在捲動畫面中的水平位置，這地方游標（cursor）的位置將標明爲閃爍的型式。

結果：

在目前游標（cursor）位置（BASL），（CH）的螢幕字元（screen character）將被設定爲閃爍型式

直到按下一個鍵爲止。

如果此Escape 鍵被偵測出，則適當的程式將被叫用，以處理所要求的功能。

游標右移箭頭鍵（控制－U）將被交回給叫用程式（calling program），而不是螢幕上位於游標（cursor）處的內容。

游標左移鍵（控制－H）將被交回到叫用程式。

從鍵盤讀來的字元將不會貯放在$φ2φφ區域。

當讀了字元之後，此在游標處的閃爍情形將被關掉，但剛讀進來那個鍵並不會被回映到螢幕上，且此游標（CH）亦不會向前移動。

當資料不貯存在$φ2φφ 區域時，刪掉輸入列（控制－X）的服務，並沒有被定義。

由於監督程式的KEYIN 常規其餘部份並沒有被叫用，故並不對 carriage return 做特別的動作。當打入一個 carriage return時，由叫用程式決定適當的動作。

返回時：

A 暫存器含著按下的鍵值。

X 暫存器含著CH 的內容。

X 暫存器並不受此常規叫用的影響。

CV , CH , BASL , H 僅當 Escape 鍵功能被利用時才會改變。

RDKEY　經由KSWL讀一個字元：放在A暫存器中回到叫用者。

組合：

X暫存器，Y暫存器和A暫存器是無關的。

CV和BASL,H是一樣的，指向捲動畫面中某列。

CH指出游標以閃爍情況顯示的水平位置。

結果：

在螢幕上游標位置的字元被設定爲閃爍。

KE IN常規透過（KSWL ）而得到控制權，以便眞正讀取鍵盤的資料。

在KEYIN程式中的返回（RTS)指令，是把控制權交回RDKEY的叫用者，而不是回到RDKEY常規。

返回時：

A暫存器含著從鍵盤讀來的字元。它可以是任何字元，包括Escape, carriage return, 右向或左向的箭頭，或者其它的控制字元。

X暫存器無改變。

Y暫存器含著CH的內容。

在螢幕區域內游標（cursor）位置處的字元已經被重新放回，即是以它被RDKEY 設定爲閃爍狀態之前的值放回的。

CV是被用來計算出新的列位置。

BASL,H存着這個再計算出的位址。

CV保留不變。

KEYIN　從鍵盤讀一個字：此字放在A暫存器中返回到叫用者。

組合：

X暫存器在經過這段常規中並沒有被用到，因此沒有受到影響。

A暫存器輸入到這段常規的東西將被貯存到螢幕區域中，游標（cursor）的位置（BASL），Y，以便在一個鍵被按下後，除掉此閃爍狀態。

Y暫存器被設定，以便將A暫存器貯存到螢幕區域內，而將在（BASL），Y處的閃爍除掉。

CH和CV 是沒有被參考到，但是必須被適當地設定。BASL，H的使用與上面描述A暫存器和Y暫存器相同。

結果：

當返回到叫用者，僅有A暫存器的值已經被改變。它含著從鍵盤暫存器而來的輸入。

KEYIN常規的取代

在很多情況下，希望以一程式來取代實際地鍵盤輸入程式，此程

式可以從鍵盤來讀取資料且預先處理輸入的東西，或者由鍵盤外的其他來源取得資訊，以供給某讀取程式。如此一個取代KEYIN 常規的程式所要的需求如下說明。如何將此程式（或常規）變成有效，是經由把進　點儲存在KSWL,H中完成的。

這個取代常規應該管理下面所指出的資源（resources）。

A-暫存器　把A暫存器儲存於（BASL）,Y，然後由所用來源處取資料存入A暫存器。

X-暫存器　必須不能改變。如果此代用程式必須使用到X暫存器，則必須在進入時保留起來而離開時重新放回。

Y-暫存器　如同上面A暫存器的使用。

當返回時，原來在進入時的內容不能改變，所以如果要使用此值的話，它必須被保留和放回。（如果輸入的來源能防止Escape 鍵和右向箭頭被輸入，則這個安全措施是不需要的。在如此的情況，可以不管Y暫存器的內容。）

CH
CV
BASL,H　這些都是用來回映（echoing）鍵盤的輸入，所以取代常規應該不管它們亦或以一適當的方法來使用它們。

註解：在取代KSWL,H處指向KEYIN的指標（pointer）時，較安全的做法爲在擺入你常規的位址之前，要先取出KSWL,H目前的

值，且貯存到一個保留區域，然後當這個取代常規工作完了時，再從保留區域中取回KSWL,H的值。

註解：如果你正在使用 DOS 時，而以你程式的定址來取代KSWL,H的內容，期望不會產生不期待的值。 DOS 使用CSWL,H 和KSWL,H 且週期性地將它們貯放回去，以顯示出 DOS 是不管它們目前的內容。依你的應用而定，有一個好的意見把這兩個指標（ pointer ）再存在一個暫時性的地方（ temporary basis ），所以到螢幕的回映將不經過 DOS 。但是要記住，要以最快的速度更正它們。

鍵盤輸入的監督常規
KEYBOARD INPUT MONITOR ROUTINE

在鍵盤服務中有許多點可讓使用人的程式有效地叫用，因爲它們是位於不同地方的一連串指令，且在進入點後的所有指令將被用到，此節中的位址表是依在監督程式的順序來排定，而不是依使用率的順序來排定。

注意一旦此監督程式跳到某特殊的點，所有在那個進入點之後所描述的啓動（ Initialization)動作，亦將被執行。在每個功能說明的結尾處，有個＆符號表示啓動動作。

位址表1—字元輸入

功　能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	被破壞的暫存器
下面兩種監督程式都有叫用RD-KEY以取得下一個字元放入A暫存器中。 與$9B(Escape)比較。 如果相等則BR到ESC以便等待下一個字元的輸入和執行Escape功能。否則的話，RTS。	FD35	64821	-715	RDCHAR	A,Y
把螢幕上游標位置設爲閃爍，原來字元保留在A暫存器，由(BASL),Y取得	FDφC	6478φ	-756	RDKEY	A,Y
&經由(KSWL)間接跳至KEYIN。	FD18	64792	744		A

功能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	被破壞的暫存器
當逐一問答(polling)鍵盤暫存器時，在位於RNDL,H的隨機數值遞增。	FD1B	64795	-741	KEYIN	A
貯存器A暫存器到(BASL),Y處(清除，由RDKEY常規所設定的閃爍)。	FD26	648φ6	-73φ		
&將鍵盤暫存器中的值放到A暫存器，清除掉鍵盤的觸發信號(strobe)然後RTS。	FD28	648φ8	-728		A
刊用在A暫存器內的字元和carry的設定，而跳到下列常規以作到Escape 鍵的服務。	FC2C	64556	-98φ	ESC1	A,Y

ⓐ HOME 清除捲動畫面(scroll window)
A ADVANCE 游標(cursor)往右
B BS 游標往左
C LF 游標下移一列
D UP 游標上移一列

功　能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	被破壞的暫存器
E CLREOL	到本列結束均清除				
F CLREOP	到畫面結束均清除				
其它	忽略：RTS				
設定埠 ϕ（鍵盤）爲輸入埠	FE89	65161	-375	SETKBD	A,X,Y
※僅適用於舊型監督程式					
叫用RDKEY以執行Escape鍵&的服務	FD2F	64815	-721	ESC	A,Y
叫用ESC1（以字元放在A暫存器內和設定carry，以便做輸入字指示的功能。返回是到RDCHAR	FD32	64818	-718		A,Y
僅適用於自動開始的監督程式					
叫用RDKEY以執行Escape鍵的服務 &	FD2F	64815	-721	ESC	A,Y
叫用ESCNEW（字元放在A暫存器內和設定carry），以便做輸入字指示的功能。	FD32	64818	-718		A,Y

功能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	被破壞的暫存器
從Escape 處理返回到RD-CHAR。					
設定 carry 旗號（flag）且JMP 到 ESC1 以處理Escape鍵功能A,B,C,D,E,F	FB97	64407	-1129	ESCOLD	A,Y
處理Escape鍵功能Ⅰ,J,K,M。轉換成D,B,A,C且叫用ESCOLD。然後到RDKEY以讀取下一個字元和降到ESCNEW 以便繼續Escape鍵的處理。	FB9B	64411	-1125	ESCNOW	A,Y
Escape 鍵處理進入點。如果A暫存器含著Ⅰ,J,K或M則跑到ES-CNOW，作轉換和處理，然	FBA5	64421	-1115	ESCNEW	A,Y

功　能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	被破壞的暫存器
再返回ESCNEW。否則的話跑到ESCOLD去處理這個輸入並離開Escape模式。					
BASL,H	4φ-41	$28-$29			
KSWL,H	56-57	$38-$39			

位址表2—列輸入

合理的說，底下應由 GETLNZ 開始談，但是在這裡所介紹的是依在監督程式裡頭指令的順序，因爲時常使用"跳至 fall into "下一個碼單元（ next code segment ）。

注意一旦此監督程式是跳到某個特定點，在那個進入點之後所描述的所有啓動動作亦將被執行。這將在每一個功能描述的後面由&暗示。

功　能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	被破壞的暫存器
經由COUT將由IN,X而來的鍵盤輸回映到螢幕上，先將INVFLG暫時	FD3D	64829	-7φ7	NOTCR	A

功能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	被破壞的暫存器
設定爲 ϕFF & 從 IN,X 中拿出一 字元。 如果是 $88 跑 到 BCKSPC。 如果是 $98 跑 到 CANCEL。 如果 X 暫存器 （輸入指示 input index ）大於 $F7 ，跳至 FD5C 否則的話跑到 NOTCR1，而略 過 Bell 常規。	FD4D	64845	-691		A
如果 X 的內容指 出已有 248 + 個輸入字元時 則響鈴。 &	FD5C	6486ϕ	-676		
遞增 X 暫存器； 如果 X 不爲零 跑到 NXTCHAR 如果 X = ϕ 跳 至 CANCEL。	FD5F	64863	-673	NOTCR1	X
將 $DC(\)放到 A 暫存器內。 以 "\"符號表 示此列輸入被 消掉。 &	FD62	64866	-67ϕ	CANCEL	A,X,Y
叫用 COUT 以印 出 A 暫存器	FD64	64868	-668		

功　能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	被破壞的暫存器
然後跳至GETLNZ。 &					
經由COUT印出carriage return。 &	FD67	64871	-665	GETLNZ	A,X,Y
將PROMPT放入A暫存器。 &	FD6A	64874	-662	GETLN	A,X,Y
叫用COUT以印出A暫存器。 &	FD6C	64876	-66φ		
將X暫存器放入$φ1，以便經過backspace動作。	FD6F	64879	-657		A,X
如果X＝φ跑到GETLNZ以便再度從GETLNZ開始。否則的話，X暫存器遞減且降至NXTCHAR。	FD71	64881	-655	BCKSPC	A,X,Y
叫用RDCHAR以取得下一個字。如果收到的字爲ctrl-U（$95，右向箭頭）則從（BASL），Y中拿出螢幕上的字元，以便	FD75	64885	-651	NXTCHAR	A

功能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	被破壞的暫存器
放在A暫存器內。					
& 如果A暫存器內容大於$DF，則與$DF作AND運算，以便將它轉換成大寫體。	FD7E	64894	-642	CAPTST	?A
& 將A暫存器貯存到輸入區域IN,X處。 與carriage return比較。 如果不等，跳至（上面的）NOTCR。 否則的話，叫用CLREOL以清除本列的剩餘部分，然後經由COUT印出carriage return，利用CLREOL的RTS功能，以完成返回到鍵盤輸入的叫用者。	FD84	649φφ	-636	ADDINP	

IN　　=$φ2φφ，鍵盤輸入區域。
INVFLG　　是在$32（5φ）。

概要—文字輸出到螢幕上
OVERVIEW—TEXT OUTPUT TO THE SCREEN

在監督程式中負責文字輸出到螢幕上的最高層面的支助爲捲動的裝置。除此，此監督程式含有很多元件，負責支援使螢幕是在格式化的型式來使用。因爲目前有很多種方法可以把文字寫到螢幕上，螢幕輸出這一個主題，已經被區分成下列章節（ sections ）：

文字輸出到捲動畫面內（TEXT OUTPUT WITHIN THE SCROLL WINDOW）

說明文字輸出的正常型態，在節零頁（ page zero ）中定義的欄位用來控制這個功能，在下面章節中描述用到的欄位。

螢幕格式控制（SCREEN FORMAT CONTROL）

依照那一種顯示作業（完全文字，完全圖形，文字與圖形的混合），捲動畫面的設定，和字元顯示模式（黑底白字，白底黑字，或者閃爍），以組合或修改進入點。

捲動畫面的資料處理（SCROLL WINDOW DATA MANIPULATIONS）

描述監督程式的一些可叫用常規（ calls）；它們是清除全部或部份捲動畫面，設定畫面的某些部份爲使用者所規定值（ specified value ）或者引發有條件或無條件地捲動畫面。

游標位置的控制（CURSOR POSITION CONTROL）

描述移動游標（相對於它目前位置）的方式和意義，或是把游標移至"與目前位置無關的某位置"。

一般文字送到螢幕（GENERAL TEXT TO THE SCREEN）

描述監督程式的進入點，以便將使用者程式所產生的資料輸出到螢幕上；或者假如CSWL被修改了，則輸送到目前的輸出裝置而且說明一些進入點，以傳遞標準型態的輸出（空白，bell碼，carriage return）到輸出裝置（一般爲螢幕）。

不用捲動畫面的文字輸出
（TEXT OUTPUT WITHOUT THE SCROLL WINDOW）

描述一些進入點，這些是用來將字元放在螢幕上而不在捲動畫面內，而且當回映到捲動畫面時，執行鍵盤的讀取。

次顯示區域（SECONDARY DISPLAY AREAS）

描述使用次文字區域的各種方法，甚至包括有限制的捲動功能，如允許鍵盤輸入反映到次顯示區域。

任合適合於一項功能以上的進入點，將出現在每一個適當的位址表中。

輸出到捲動畫面
OUTPUT WITHIN THE SCROLL WINDOW

捲動畫面的作業與印字機（printer）或打字機（type writer）

的輸出很相似，其間新字是在前一個輸出字的右邊顯示，且新列是被顯示在前一線的下面。本節所描述的就是這種作業模式；即是，本節描述如何依CSWL向量（vector）的意思，即將信息印到螢幕上或者印到一個印字機型式的裝置。在"一般文字送到螢幕"（General Text to the Screen）單元中描述螢幕的使用，略過CSWL向量而直接使用捲動畫面的輸出常規。

在Apple II中所提供"顯示輸出資訊"的正常方法——將每個要被顯示的字或者格式控制的字元（如一個carriage return）放在A暫存器內而叫用COUT。在COUT中，透過CSWL向量作一個間接跳越，以跳至一個常規，此常規將字元放在所選定介體（medium）上或者完成此字元所指的控制功能。當此系統被啓動後，這個向量被設定爲指向COUT 1，此常規爲負責將捲動畫面輸出送到螢幕上。如果使用者設定一個不同的輸出裝置（在BASIC中使用PR#n 或者在監督程式模式中使用ctrl-P），則此CSWL 向量將被設定爲將輸出的位元組，送到所選用的週邊控制器卡（peripheral controller card），而不送到螢幕上。依那一種型式的週邊控制器卡，而那種控制型式爲有效，在那個卡中的程式，可能將字元放在輸出裝置上，然後亦會跳到COUT1 以便也將它寫到捲動畫面上。

文字輸出到螢幕的一般模式爲在"捲動（scroll）"模式。在這個模式中，新的資訊是被寫到螢幕的底列，而且螢幕的內容往上移，往上移以便讓新資訊出現在舊資訊之下。這種輸出模式是使用在APPLESOFT或者BASIC的" PRINT"指令。任何把資料顯示到螢幕上

的監督程式命令均採用此種輸出模式。

當新的字元要被寫到螢幕上時，它們是被放在游標的位置。游標位置是一個螢幕上的位置（和在螢幕的再充電記憶體（refresh memory）中），是由在第零頁內某些欄位的內容指明。捲動畫面是螢幕的一部份（或者全部），也是由第零頁內某些欄位的內容中定義。沒有特殊地顯示器硬體是與捲動功能有關的。在監督程式中需要一些常規用來移動螢幕再充電記憶體（screen refresh memory）內資料以支援此捲動功能。

在第零頁中有些欄位用來描述此捲動畫面，指出左限的行數和寬度，和上頂及下底，我們將在此描述。

游標的位置在不同的單元中被定義，除非使用者的程式把它弄亂，否則這些欄位是互相通用的。

游標（cursor）位置所在的螢幕列號碼是存在 CV 欄內。CV 指出游標相對於螢幕頂列（top line）的列號碼，而不是相對於捲動畫面（scroll window）。（注意這與下面所描述的CH不同。）對應到這個列號碼的螢幕再充電記憶體之位置被保留在兩個位元組（BASL, H）欄位內。因此假如捲動畫面（Scroll Window）的左邊邊緣並不是螢幕上的最左邊字，則BASL, H將被調整以指向捲動畫面內，在那列最左邊字元的位置。所以，一個程式可能尋向CV以便決定出游標的列號碼，但是此程式不能僅把一個不同的列號碼POKE到CV 中以

移動此游標（cursor），而必須連BASL,H也一起更改。

游標的水平位置被保留在CH中。在CH內的值是相對於捲動畫面（Scroll Window）的左邊邊緣，而不必要相對於螢幕邊緣。當一個字元要被"寫"或者"印"到螢幕上時，將字元放入螢幕的再充電記憶（screen refresh memory）中的那個常規，使用Y暫存器爲水平位置，它是假定已經將CH的值放到Y暫存器中。在位址表中，每一條說明指出被叫用的常規是使用CH或者Y暫存器。

對於機器語言的程式，捲動畫面的輸出可以輕易的完成——將輸出位元組放入A暫存器內而用一個JSR跳到在$FDED(-531)處的COUT常規。在BASIC中要完成相同的事，可以印出一個變數（variable），變數內存著要印出的位元組。當然啦！在BASIC中，用一個命令就可以寫出一整串的字串。

當這些字元傳通過COUT1時，如果有必要，根據名爲INVFLG欄的內容，可把字元修改，而寫成黑底白字，白底黑字或閃爍。這個欄位可以在任何時間以（POKE）來設定，且此馬上有效地影響到所有未來要經由程式印出去的字，直到它再度地被修改爲止。這個功能僅應用到程式的印字輸出，而在鍵盤輸入時，當每個輸入字要經由COUT被回映在螢幕時，INVFLG將暫時被更改成$FF。

BAS2L,H這兩個位元組欄位對使用者程式沒有什麼用途，它僅是一個在捲動作業中使用的工作區而已。

第零頁的欄位 PAGE ZERO FIELDS

十進位	十六進位	監督程式	說　　明
32	$2φ	WNDLFT	捲動畫面的左邊行位(Left column)： 範圍是φ到39（$27）。 本欄僅在VTABZ 用到。其內容（當被使用者程式改變後）在下一個捲動動作，消除到頁的結尾或carriage return 輸出時，才有用到。 CH含著游標相對於（WNDLFT）的水平位置。 在改變了WNDLFT 的內容之後，再CALL VTAB或者印一個carriage return到螢幕時，將使WNDLFT產生作用。
33	$21	WNDWDTH	捲動畫面(Scroll Window)的寬度： 範圍是1到40-(WNDLFT)。 當一個字元經由COUT寫到螢幕時，它是被放在(BASL)，(CH)處，做完了此動作後CH加1。然後(CH)與(WNDWDTH)比較以判斷游標(cursor)是否已經超過了

十進位	十六進位	監督程式	說　　明
			捲動畫面的右邊邊界（right margine）了。
34	$22	WNDTOP	捲動畫面的頂列（top line）： 對完全文字螢幕而言，範圍是0到22（$16）。 對圖形與文字混合時而言，範圍是2φ到22（$14到$16）。在一個捲動動作時，本欄用來指出捲動動作應該從何處開始。
35	$23	WNDBTM	捲動畫面的底列＋1： 範圍是（WNDTOP）+1到24（$18）。 WNDBTM 指出捲動畫面下面第一列的列號。WNDBTM 的內容僅當輸出一個 carriage return（$80）或者 line feed（$8A）時才被測試。它是被"清除到頁的結束"（Clear to End of Page）和捲動常規所使用。
36	$24	CH	距WNDLFT 的位移，此處爲下一

十進位	十六進位	監督程式	說　明
			個送到螢幕的字元要放置的地方：範圍是 ϕ 到（WNDWDTH）—1 。在螢幕輸出常規 STOADV 將一個字元放到螢幕區域，以當做正常字元輸出的一部份之後，CH 加1 且與WNDWDTH 比較。如果 CH 不比 WNDWDTH小時，則一個 carriage return 將被模擬產生。注意 CH 是用被 GETLN 等常規使用，以把鍵盤輸入回映在螢幕上（因爲使用了 COUT ）。
37	$25	CV	下一個字元被寫到螢幕上垂直位置（列號 line number ）：範圍是 ϕ 到 23（$17）。CV 的內容是相對於螢幕的頂端，而不是捲動畫面的頂端，這個值可以經由"把希望的列號碼載入 A 暫存器且叫用 TABV "而設定；也可以經由（ pokeing ）把列號放入 CV ，然後叫用 VTAB 而設定。一個字元眞正貯存到螢幕區域是使用 BASL,H以得到列號，而不是使用

十進位	十六進位	監督程式	說　　明
			CV。上面叫用VTAB或TABV是根據CV來設定BASL,H，以便能馬上供未來參考。 如果CV是在WNDBTM上或者位於WNDBTM下方，在carriage return經過時，CV仍維持在目前螢幕列上，而捲動畫面的內容將每次往上捲。
4ϕ	$28	BASU	這兩個位元組的欄位是個記憶體位
41	$29	BASH	址，它是在捲動畫面內，目前文字列左端字元的位置。其內容爲CV和WNDLFT的一個函數。 本欄是由BASCALC常規來設定，以指出“A暫存器內所指的螢幕列的左端字元的記憶體位址”。叫用BASCALC通常是在VTAB常規內完成，然後VATB常規將(WNDLFT)加到BASL,H，以便指向在捲動畫面內列的左邊位置。
42	$2A	BAS2L	這兩個位元組僅在一個捲動動作時

十進位	十六進位	監督程式	說　明
43	$2B	BAS2H	，方被當做一個工作區域。它是一個目的地列的指標，在每一列往上移時使用。
5φ	$32	INVFLG	這個位元組爲一面罩信號（mask），由COUT1 使用造成要寫到螢幕區域上的字，是在黑底上顯示白字（INVFLG=$FF）或者在白底上顯示黑字（INVFLG=$3F）或者閃爍（blinking）（INVFLG=$7F）。當RESET 發生時，在SETNORM內的常規會把此欄定爲 $FF。此名爲 SETINV的常規可被叫用以設定相反的視頻（reverse video）。這個監督程式自己不會設定閃爍畫面。 注意：INVFLG=$7F 並不會造成所有的字閃爍：此大寫Z字的位元必須爲 φ1 以便閃爍能發生。
53	$35	YSAV1	在螢幕輸出常規被召用時，此位組元組是Y暫存器的儲存區。Y暫存器是在COUT 1 常規被保留和重新取回。

十進位	十六進位	監督程式	說　　明
54	$36	CSWL	這兩個位元組欄位存著"輸出字元的接收和處理常規"的位址。當按下RESET 時，這個欄位被設定新值以指到COUT1 ，以把輸出字元送到螢幕上。打入一個監督命令nPc（n=埠號碼，Pc =控制－p control-P ）將使監督程式把CSWL,H定爲Cnφφ。本欄的指的常規將接收每個經由COUT "寫入"的位元組（在A暫存器內），命令nPc 可視爲一個JMP（CSWL）。 如果執行此監督命令"φPc"，則CSWL,H被設定爲指向COUT1，而不指向Cφφφ 。
55	$37	CSWH	

捲動畫面的輸出常規　SCROLL WINDOW OUTPUT ROUTINES

功　能	十六進位位址	正十進位位址	負十進位位址	監督程式的標誌	破壞的暫存器
經由CSWL 的間接跳越，印出字。	FDED	65φφ5	-531	COUT	none
利用INVFLG，將位於A暫存	FDFφ	65φφ8	-528	COUT1	none

功能	十六進位位址	正十進位位址	負十進位位址	監督程式的標誌	破壞的暫存器
器內的位元組寫到螢幕上游標(CV),(CH)處，並移動游標。					
有移動游標，但不用INVFLG，將位於A暫存器內的位元組寫到螢幕上位於(CV),(CH)處。	FDF6	65φ14	-522	COUTZ	none
經由COUT印出carriage return。	FD8E	6491φ	-626	CROUT	A
經由COUT印出"ERR"和鈴聲碼。	FF2D	65325	-211	PRERR	A
經由COUT印出鈴聲碼($87)。	FF3A	65338	-198	BELL	A
用CV(和WNDLFT)來設定BASL,H。	FC22	64546	-99φ	VTAB	A
用(A)和WNDLFT來設定BASL,H而與CV無關。	FC24	64548	-988	VTABZ	A
設定BASL,H指向A暫存器所指螢幕列(	FBC1	64449	-1φ87	BASCALC	A

功　能	十六進位位址	正十進位位址	負十進位位址	監督程式的標誌	破壞的暫存器
而不是畫面列）的左端（left end）。					
CH	36	$24	WNDLFT	32	$2φ
CV	37	$25	WNDWDTH	33	$21
GBASL,H	38-39	$26-27	WNDTOP	34	$22
BASL,H	4φ-41	$28-29	WNDBTM	35	$23
INVFLG	5φ	$32			

由常規控制的螢幕格式
SCREEN FORMAT CONTROL BY ROUTINE

這一個表指出在監督程式中那些地方是用來控制顯示模式和捲動畫面的架構。

功　能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
清除掉HIRES的圖案模式。	& FB33	643φ7	-1229		A
設定主要顯示區域。	& FB36	6431φ	-1226		A
設定TEXT模式。	& FB39	64313	-1223	SETTXT	A
將φ放到A暫存器內以供WNDTOP用，跳至下面的SETWND	& FB3C	64316	-122φ		A

功能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
設定圖形模式。	& FB4φ	6432φ	-1216	SETGR	A,Y
設定圖形與文字混合的模式。	& FB43	64323	-1213		A,Y
叫用CLRTOP以清除圖案。	& FB46	64326	-121φ		A,Y
將2φ(＄14)放到A暫存器內以便設定WNDTOP。降到SETWND。	& FB49	64329	-12φ7		A
拿A暫存器值來設定畫面的頂線(WNDTOP)，其值爲φ或2φ或由使用者設定。經由下面繼續執行。	FB4B	64331	-12φ5	SETWND	A
放φ到A暫存器以設定WNDLFT。	& FB4D	64333	-12φ3		A
將A暫存器存放到WNDLFT。	& FB4F	64335	-12φ1		A
放4φ到A暫存器以設定WNDWDTH。	& FB51	64337	-1199		A
將A暫存器存放到WNDWDTH。	& FB53	64339	-1197		A
放24到A暫存器	& FB55	64341	-1195		A

功　能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
以設定WND-BTM。					
將A暫存器放到&WNDBTM。	FB57	64343	-1193		A
放23到A暫存器&以供VTAB 使用。	FB59	64345	-1191		A
貯存A暫存器到&CV。	FB5B	64347	-1189	TABV	A
跳到VTAB —設定BASL,HRTS。					
放$FF到Y暫存器以設定IN-VFLG降至SETIFLG。	FE84	65156	-38φ	SETNORM	Y
放$3F到Y暫存器以設定IN-VFLG。BR到SETIFLG	FE8φ	65152	-384	SETINV	Y
將Y暫存器存入INVFLG 且RTS。 $FF黑底白字（從SETNORM）。 $3F白底黑字（從SETINV）。	FE86	65158	-378	SETIFLG	none

功能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
$7F 閃爍（從使用者叫用而來的字，其較高位的兩個位元爲 ϕ1）					
設定CSWL,H以指向COUT1。	FE93	65171	-365	SETVID	A,X,Y

CH	36	$24	WNDLFT	32	2\phi$
CV	37	$25	WNDWDTH	33	$21
INVFLG	5ϕ	$32	WNDTOP	34	$22
BASL,H	4ϕ-41	$28-29	WNDBTM	35	$23
CSWL,H	54-55	$36-37			

經由POKE/STORE控制螢幕的格式
SCREEN FORMAT CONTROL BY POKE/STORE

在很多情況下，在前一頁中所描述的位於監督程式內的常規是存在的，因爲監督程式本身使用此描述過的功能。時常，叫用監督程式以得到一特殊的控制功能是一種很難的方法。這個表指出要完成相同的結果的其它的方法。

功能	方法
設定GRAPHICS顯示模式。	POKE -163ϕ4,ϕ 或 STA Cϕ5ϕ
設定TEXT顯示模式。	POKE -163ϕ3,ϕ 或 STA Cϕ51
設定GRAPHICS 模式爲完全螢幕。	POKE -163ϕ2,ϕ 或 STA Cϕ52
設定GRAPHICS和TEXT 的混合模式。	POKE -163ϕ1,ϕ 或 STA Cϕ53

功　能	方　法
設定顯示主要區域(Primary Area)。	POKE -163φφ,φ　或　STA Cφ54
設定顯示次要區域(Secondary Area)。	POKE -16299,φ　或　STA Cφ55
針對 graphics, 消除HIRES/設定LORES。	POKE -16298,φ　或　STA Cφ56
設定HIRES Graphics 模式。	POKE -16297,φ　或　STA Cφ57
設定捲動畫面的頂列(top line)。	POKE 34, 列號碼 (φ - 23) 底列必須大於頂列。
設定捲動畫面的左邊邊緣。	POKE 32, 行號碼 (φ - 39) 左邊邊緣＋寬度不超過 40 。
設定捲動畫面的寬度。	POKE 33, 行號 (1-4φ) 左邊邊緣＋寬度不超過 40 。
設定捲動畫面的底列。	POKE 35, 列號碼 (1-24) 底線必須大於頂線。
設定一般 (黑底白字上) 文字。	POKE 5φ,255 或者把 $FF 存入 $32 中。
設定閃爍的文字。	POKE 5φ,127 或者把 $7F 存入 $32 中。
設定反相 (白底黑字) 文字。	POKE 5φ,63 或者把 $3F 存入 $32 中。

如果是使用上列的方法來改變捲動畫面的架構，使用者的程式亦應該想辦法以保證在畫面內，游標 (cursor) 是在一個妥當的位置 (CV, CH, BASL, H)。CALL-936將會把游標放在畫面內。

C\phi$5$\phi$ 和 C\phi$51 控制了文字模式 / 所有或部份圖形。其他項目關於HIRES 或LORES或完全或部份螢幕畫面可以先被建立起來，但是直到 C\phi$5$\phi$ 被設定以前，是不會明顯的。同樣地， C\phi$51將不管其它的設定而回到文字模式（ Text Mode ）。

捲動畫面的資料處理
SCROLL WINDOW DATA MANIPULATIONS

這個表說明了三種關於捲動畫面資料處理的進入點。第一個爲監督程式內標誌爲 ESC1（ Escape 鍵的處理器），因爲它依 A 暫存器的內容和 carry 的被設定而將控制權傳遞給其它的進入點，包括自動開始的監督程式的一個進入點，因爲它處理 ESC1 的一個需求 —— carry 被設定的情況。

此表的第二部份爲一堆進入點，它們是用來清除或設定部份的螢幕爲一特定的值。

此表的第三部份爲描述造成有條件或無條件地畫面捲動的點。

位址表

功　能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
叫用螢幕資料處理。 如果Carry被設定且A暫存器＝ @跑到 HOME A跑到 ADVANCE B跑到 BS C跑到 LF D跑到 UP E跑到 CLREOL F跑到 CLREOP 其它 RTS至叫用者 在這些功能結束的RTS，將控制權交回到ESC1的叫用者。	FC2C	64556	-980	ESC1	A,Y

功　能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
從列(CV)行(CH)處開始清除到捲動畫面的結尾處。	FC42	64578	-958	CLREOP	A,Y
從列(CV)行(Y)處開始清除	FC44	6458ϕ	-956		A,Y

功能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
到捲動畫面的結尾處。					
從列(CV)行(Y)處開始清除到捲動畫面的結尾處。	FC46	64582	-954	CLEOP1	A,Y
清除捲動畫面爲空白，游標設定在畫面的左上角。	FC58	646φφ	-936	HOME	A,Y
設定CH=φ,CV=(A)，清除到EOP(end of page=end of window)。	FC5A	646φ2	-934		A,Y
從列(A)處開始清除到空白處，游標設定在列(CV)的左端	FC5C	646φ4	-932		A,Y
從游標(cursor)((BASL),(CH))處清除至列結束。	FC9C	64668	-868	CLREOL	A,Y
從游標(BASL),Y處開始清除列。	FC9E	6467φ	-866	CLEOLZ	A,Y
從游標(BASL),Y到列結束，設定爲位於A暫存器內的字	FCAφ	64672	-864	CLEOL2	A,Y

功　能		十六進位位　址	正十進位位　址	負十進位位　址	監督程式標　誌	破壞的暫存器
清除列(BASL)，然後用CV和WNDLFT來設定BASL，H。		FC95	64661	-875	SCRL3	A,Y
從游標(BASL)，Y處清除此列，然後用CV & WNDLFT來設定BASL，H。CH保持不變。		FC97	64663	-873		A,Y
零放入A暫存器內以設定CH。	&	FC62	6461φ	-926	CR	A,?Y
將A暫存器存入CH中。	&	FC64	64612	-924		A,?Y
CV加1。	&	FC66	64614	-922	LF	A,?Y
比較CV和WNDBTM。如果(CV)＜(WNDBTM)，設定BASL，H，如果需要的話，做捲動動作。		FC68	64616	-92φ		A,?Y
捲動畫面從列(CV)到列(WNDBTM)。		FC7φ	64624	-912	SCROLL	A,Y

功能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
捲動畫面從列(A)到列(WNDBTM)。	FC72	64626	-91φ		A,Y
自動開始監督程式擴充的服務(Autostart Monitor extended service)					
設定Carry旗號且JMP到ESC1以便處理Escape鍵功能A,B,C,D,E,F。	FB97	644φ7	-1129	ESCOLD	A,Y

CH	36	$24	WNDLFT	32	$2φ
CV	37	$25	WNDWDTH	33	$21
INVFLG	5φ	$32	WNDTOP	34	$22
BASL,H	4φ-41	$28-29	WNDBTM	35	$23

游標的位置控制
CURSOR POSITION CONTROL

一般而言，游標所在的位置由CV（相對於螢幕頂上的列號碼）的內容和CH（相對於捲動畫面左邊邊緣的行號碼）的內容來指出。游標所在的記憶位置爲BASL,H（此含著在捲動畫面內，某列最左邊字的位址）內容與CH內容之和。通常BASL,H含著依CV和WNDLFT內容計算而來的一個位址。但是假若CV或WNDLFT 被更改而

沒有再計算BASL,H的值，則監督程式的不同常規可能被執行而產生不可預測（至少不是所希望）的結果。

在下面的表中，此描述包括指出那一個游標位址欄有什麼作用。注意如在$FC95處，將BASL,H所指的那一列被清除，然後用CV,WNDLFT重新計算BASL,H以供未來參考之用。

ESC1和VIDOUT 常規亦包括在此表中，因爲在可以在進入傳給它們適當的A暫存器內容，便可使用（goto）到其它進入點。VIDOUT是個處理CR, backspace , line feed的常規，當這種字元經由COUT 1要送出去時，即由VIDOUT加以處理，ESC 1是當鍵盤常規是以ESCAPE鍵模式操作時，被叫用以完成所希望的功能。所以，它有4種游標（cursor）移動方向的能力，和具有清除捲動畫面由游標目前的位置至此列的結尾處或至捲動畫面的結尾處。亦或清除整個捲動畫面和將游標（cursor）放在左上角地方的能力。所要執行的功能是依在進入時A暫存器的內容和此Carry處理器狀態位元的"設定"情況而定。

自動開始監督程式的進入點ESCOLD，由於它與ESC1的關係，所以被包括在表中。

下一組的點包含著那些用來清除在螢幕上的資料和移動游標。

第三組是一些進入點，用來負責游標（cursor）相對於它目前

位置的移動。

第四組爲負責將游標（cursor）放在一所希的位置，而不參考到它目前的位置。爲了達到這目的，此程式應該設定CV和CH，然後叫用VTAB以設定BASL,H。

位址表

功能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
叫用screen/cursor處理。 如果carry被設定且A暫存器= @跑至 HOME A跑至 ADVANCE B跑至 BS C跑至 LF D跑至 UP E跑至 CLREOL F跑至 CLREOP 其它的 RTS到叫用者。 在這些功能中每一個結尾處的RTS,將控制權交回給ESC 1的叫用者。	FC2C	64556	-98φ	ESC1	A,Y
設定Carry旗號	FB97	644φ7	-1129	ESCOLD	A,Y

功　能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
且JMP到ESC 1以處理Escape鍵的功能A,B,C,D,E,F。		僅適用於自動開始的監督程式			
將字放在螢幕記憶體中或者處理控制字元。	FBFD	645φ9	-1φ27	VIDOUT	A,Y
如果(A)＞$9F或者＜$8φ跑至STOADV。 如果(A)＝$8D跑至CR。 如果(A)＝$8A跑至LF。 如果(A)＝$88跑至BS。 如果(A)＝$87響"鈴聲"。 如果(A)＝其它，忽略它；RTS					
清除捲動畫面，設定游標(cursor)到畫面的左上角。	FC58	646φφ	-936	HOME	A,Y
設定CH=φ,CV=(A)，清除到EOP(end of page＝end of window)	FC5A	646φ2	-934		A,Y
清除畫面範圍從列(A)到空白，游標(cursor)設在列(CV)的左端。	FC5C	646φ4	-932		A,Y

功能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
清除列(BASL)，然後以CV和WNDLFT來設定BASL，H。	FC95	64661	-875	SCRL3	A,Y
從游標(BASL)，Y處清除此列，然後用CV&WNDLFT來設定BASL,H。	FC97	64663	-873		A,Y
將CH放入Y。	& FBFφ	64496	-1φ4φ	STOADV	A,Y
將A暫存器貯放到位於(BASL),Y的螢幕處	& FBF2	64498	-1φ38		A
CH加1。	& FBF4	645φφ	-1φ36	ADVAMCE	A
比較(CH)與(WNDWDTH)如果CH不比它小，跑至CR。否則的話返回(RTS)。	& FBF6	645φ6	-1φ34		A
游標向左移一格，如果需要且(CV)＜(WNDTOP)則移到前一列的右端。	FC1φ	64528	-1φφ8	BS	A

功　能		十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
如果(CV)＜(WNDTOP)則游標向上移一列。		FC1A	64538	-998	UP	A
將零放到A暫存器以便給CH用。	&	FC62	6461φ	-926	CR	A,?Y
A暫存器貯存到CH。	&	FC64	64612	-924		A,?Y
CV加1。	&	FC66	64614	-922	LF	A,?Y
比較CV和WNDBTM。如果CV不比它小，則CV減1並作捲動動作。如果CV較小，則跑至VTABZ以設定BASL,H和返回。		FC68	64616	-92φ		A,?Y
將閃爍游標(cursor)放在列(A)行(CH)處(將A貯存入CV，且經由跳至VTAB設定(BASL,H)。		FB5B	64347	-1189	TABV	A

功能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
經由叫用BASCALC和加上WNDLFT，以便用CV和WNDLFT來設定BASL,H。	FC22	64546	-99φ	VTAB	A
經由叫用BASCALC和加上WNDLFT以便用A暫存器和WNDLFT來設定BASL,H	FC24	64548	-988	VTABZ	A
將BASL,H設爲某記憶位址，此位址是A暫存器所指之列的最左邊字元（不是畫面的左端字元）。	FBC1	64449	-1φ87	BASCALC	A
經由CSWL作跳越，印出字元。	FDED	65φφ5	-531	COUT	?A
字元送到螢幕輸出常規的進入處——一般而言爲CSWL內含的位址。將字依適當的控制動作和控制	FDFφ	65φφ8	-528	COUT1	?A

功　能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
字的要求，印到螢幕上去。如果(A)＜$Aφ跑到COUTZ，略過反相的視頻遮蔽（inverse video mask）。					
僅適用於自動開始的監督程式（AUTOSTART MONITOR ONLY）					
經由VIDWAIT（如果操作者要求則會中止（pause））和VIDOUT而將字元印到螢幕上去。其間A暫存器和Y暫存器被保留起來和重新取回	FDF6	65φ14	-522	COUTZ	none
僅適用於舊型的監督程式（OLD MONITOR ONLY）					
經由VIDOUT而將字元印到螢幕上去，其間A暫存器和Y暫存器被保留起來和重新取回。	FDF6	65φ14	-522	COUTZ	none

CH	36	$24	WNDLFT	32	$2φ
CV	37	$25	WNDWDTH	33	$21
INVFLG	5φ	$32	WNDTOP	34	$22
BASL,H	4φ-41	$28-29	WNDBTM	35	$23

一般文字印到螢幕上
GENERAL TEXT TO THE SCREEN

把文字送到螢幕的比較好方法爲將要送出去的字放在A暫存器內，然後叫用 COUT開始處理。這方法較好的原因是因爲如果你要把輸出送到非螢幕的同樣裝置，你可改變CSWL,H以指向"負責此裝置運作的程式"。而大多時候您不管 CSWL,H 目前的設定只要把字元寫上螢幕。COUT 1 是僅作螢幕輸出的進入點，其間的反相視頻顯示（reverse video display）或者閃爍字元的設定是利用 INVFLG。自動開始監督程式由此點進入時，亦允許你使用 control-S 鍵而停止輸出。

COUTZ 亦可用作爲螢幕的輸出，而沒有用 INVFLG來修飾輸出字元。那就是，以一個字放在A暫存器內而叫用 COUTZ ，將會把那個字依樣放在螢幕上，而沒有用 INVFLG以反相的視頻或閃爍模式來顯示此字。在自動開始的監督程式中，在COUTZ 處的進入點亦足夠處理 control-S的輸入，當此鍵盤緩衝器內含著一個 control -S時，

如果被寫進來的字爲一 carriage Return，則系統將會停住。

VIDOUT 常規解釋此字元且如果它並不是一控制字元時，則將它放在螢幕上。如果此VIDOUT常規直接被叫用（舉例而言，在自動開始監督程式中略過控制S處理），則此叫用程式（calling）必須先保留此A暫存器和Y暫存器，做完之後再重新取回A暫存器和Y暫存器內，因爲它們在VIDOUT程式內均被毀壞。

輸出到螢幕上可以經由這些不同的進入點而寫入。但是，注意此監督程式仍將使用此 COUT，以達成鍵盤輸入回映的功能（keyboard input echo function），它暫時是設定 INVFLG 爲 $FF，以便將每個被回映的字寫成黑底白字。

底下是前面所提位置的位址，和一些其它的進入點，這些進入點將輸出特殊的字元（經由 COUT），而叫用程式在召用之前不必先把字元放入 A 暫存器中。

位址表

功　能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
經由 JMP（CSWL），而印出一個位元組至一特殊的輸出裝置，一般而言	FDED	65φφ5	-531	COUT	none

功能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
，COUT1爲將字放到螢幕上字元印至螢幕上的輸出常規之進入點——通常此位址存在CSWL內。以適當動作來安排控制權和控制字元，而把字元寫上螢幕。如果(A)＜$Aφ跑至COUTZ，而略過反相的視頻遮蔽（inverse video mask）。	FDFφ	65φφ8	-528	COUT1	none
把(A)和INVFLG &作AND運算。	FDF4	65φ12	-524		?A
印一個位元組到螢幕上。看本表結尾處的自動開始與舊型監督程式的不同之處。	FDF6	65φ14	-522	COUTZ	none
處理A暫存器內的字元寫到螢幕上。如果爲控制的	FBFD	645φ9	-1φ27	VIDOUT	A,Y

功　能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
字，做控制動作。 如果爲顯示的字，則放入螢幕的復新記憶體（refresh memory）內。					
把A暫存器存放到螢幕上（BASL,H），（CH）位置，然後CH加1 如果超過畫面的話，跑至CR。	FBFφ	64496	-1φ4φ	STOADV	A,Y
CH加1且如果超過畫面的話，跑至CR。	FBF4	645φφ	-1φ36	ADVANCE	A
測試CH。如果CH=>WND-WDTH跑至CR。	FBF6	645φ2	-1φ34		A
如果A=$8D，$8A,$88，或者887做: $8D carriage return跑至CR。 $8A line feed跑至LF。 $88 back space跑至BS。 $87 bell 響"鈴聲"。	FCφ4	64516	-1φ2φ		A
設定INVFLG爲	FE8φ	65152	-384	SETINV	Y

功能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
$3F=反相的視頻。					
設定INVFLG爲$FF=一般的視頻。	FE84	65156	-384	SETNORM	Y
將Y暫存器放到INVFLG。	FE86	65158	378	SETIFLG	Y
僅適用於自動開始的監督程式（AUTOSTART MONITOR ONLY）					
經由VIDWAIT（如果操作者要求停止則停住）和VIDOUT將字印到螢幕上，其間A和Y暫存器被保留和重新取回	FDF6	65φ14	-522	COUTZ	none
測試操作者的暫停要求。如果(A)=$8D（carriage return），且鍵盤暫存器有東西了，且如果鍵盤暫存器含著cntl-S，則跑至KBD-WAIT。否則的話，跑到VIDCUT。	FB78	64376	-116φ	VIDWAIT	Y

功　能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
每當操作者要求時將系統暫停。作廻路直到新鍵被按下去爲止。如果下一個按下的鍵爲cntl-C 則跑到VIDOUT，而留下cntl-C在鍵盤暫存器內。否則的話，清除鍵盤觸發信號(keyboard strobe) 且跑至VIDOUT。	FB88	64392	-1144	KBDWAIT	Y
僅適用於舊型的監督程式(OLD MONITOR ONLY)					
經由VIDOUT把字元印到螢幕上，其間有保留和重新取回A暫存器和Y暫存器的內容。	FDF6	65φ14	-522	COUTZ	none

CH	36	$24	WNDLFT	32	$2φ
CV	37	$25	WNDWOTH	33	$21
INVFLG	5φ	$32	WNDTOP	34	$22
BASL,H	4φ-41	$28-29	WNDBTM	35	$23

控制字元 CONTROL CHARACTERS

注意：

下面的控制字元對螢幕顯示有特殊意義。

$8D Carriage Return

在自動開始監督程式中，COUTZ 常規叫用（JSR）VIDWAIT，VIDWAIT 在跳至 VIDOUT 之前處理 control-S的功能。在舊型的監督程式中COUTZ 是叫用VIDOUT。

當VIDWAIT常規測出將被寫出的字元爲一Carriage Return時，它則測試鍵盤輸入緩衝器是否爲一 control-S。如果是的話，則它清除掉鍵盤的硬體，以便能打入其他的輸入，廻路等待直到其它鍵被按下去爲止。如果這個輸入並不是 control-C，則此鍵盤輸入的觸發信號被清除。否則的話，此鍵盤緩衝器內仍是 control-C，以便讓叫用程式去檢驗和處理（detect and handle）。然後VIDWAIT 跳到VIDOUT。

$8A line feed

游標（cursor）往下移一列，只有是要將它放在捲動畫面的下一列例外。在此例外情況時，捲動畫面的內容是被往上移一列，且游標是停留在目前的螢幕列上。

$88 backspace

VIDOUT 常規把 CH 減 1 而將游標向左移一格。如果 C H 變成爲負值，則 CH 被設定爲（ WNDWDTH ）－ 1 且 CV 減 1 。如果因減少 CV 而產生超過（ WNDTOP ）的情況，則 CV 將不減 1 。負向的捲動是不允許的。

$87 響鈴聲 Sound the Bell

喇叭以每秒 1ϕϕϕHz 的頻率持續三十分之一秒。

任何其它字元在 $8ϕ 到 $9F 範圍內將被丟掉。它並不造成游標（ cursor ）的移動或記憶體的更改。

不用捲動畫面的輸出
OUTPUT WITHOUT THE SCROLL WINDOW

如果所有或者部份的螢幕以直接定址方法來使用，則必須避開某些監督程式的服務。一般而言，由監督程式所提供的捲動畫面服務爲：

1. 如果一個 carriage return 或者 line feed 使得游標（ cursor ）向下移出畫面的底線，則把所有畫面內的文字向上捲一列。

2. 如果超過了畫面寬度，則自動假設一個 carriage return 。

3. 在一個 carriage return 時將游標（ cursor ）放在捲動畫面的左邊邊緣，而不是放在螢幕的左邊邊緣。

4. 提供螢幕清除功能：

A. 清除畫面，而將游標（cursor）放在左上角。

B. 從游標（cursor）目前所在的位置，清除此畫面。

C. 清除一列（由游標所在位置到列結束）。

當使用所有或者部份的螢幕爲一隨意存取（random access）顯示區時，應該避開這些自動的服務。

如果此整個螢幕被當做一個隨意存取顯示區，而不是一部份被當做一個工作捲動畫面（Working Scroll Window），這問題並不會太困難。考慮上是使得整個螢幕被定義爲捲動畫面。

1. 如果一個 carriage return 或者 line feed 或者超過畫面寬度事件在捲動畫面的底列時發生，捲動動作才會發生。爲了避免開這個動作，可以令程式在螢幕底線時不輸出 CR 或者 LF 或者不會有超長的資料；且在鍵盤輸入動作時，保持游標（cursor）不在螢幕的底列。

2. 當 RESET 鍵被按下去時整個螢幕由監督程式被定義成爲捲動畫面（Scroll Window）。如果畫面架構已因叫用在 $FB3C（64316或－122φ）的“設定正常捲動畫面”而變動，使用者程式可以再存回畫面參數以回復原來架構。

3. 在印出一串字元之前，將游標定在所希望的地方：POKE此列號

碼到CV中且叫用VTAB，然後POKE字號碼到CH中。

4. 如同操作捲動畫面一樣的方法輸出一串字，必須注意不要不小心地超過畫面寬度或者輸出 carriage returns。無論如何，依你螢幕的設計，你必須小心地做這些動作中每一個。

注意程式輸出一個 carriage return，並不會把此列上 carriage return 右邊的東西清除，但是鍵盤輸入一個 carriage return 時則會發生此種現象（如果讀鍵盤是經由監督程式的取列（get-line）常規來做的話）。

如果部份的螢幕被設定爲一操作的捲動畫面時，而螢幕的其餘部份被直接定址使用，則必須叫用一些不同（較低）層次的監督程式服務。

達到分割螢幕（divided screen）的一種方法是使用捲動畫面以監督程式的取得一輸入一列（get-input-line）服務作資料輸入，且使用此捲動畫面以支援程式要放在那裡的任何輸出。然後使用部份的 CORES 圖形負責將字元放在捲動畫面以外的螢幕上，如同下面所描述。這裡的目標爲游標位置（第零頁的CV, CH和BASL, H各欄）交給監督程式負責，且使用其它的方法／欄位以便將字元放在捲動畫面之外。

爲了要將字元放在捲動畫面之外，

1. 把列號碼放在A暫存器內，叫用GBASCALC常規設定GBASL，H以指向所指螢幕列的最左邊字元之記憶體位址。

2. 以Y暫存器指出在列上的水平位置，把所希的字元存入(GBASL),Y位置。

注意如果此螢幕被當爲一混合模式使用，這個技術並不會和LORES畫圖衝突，因爲PLOT的叫用都是依所要求的位置設定GBASL,H，而與以前可能的內容無關。

另一種方式對BASIC或者APPLESOFT程式師是適用的。同樣的，當下面的方法用來將字元放在捲動畫面之外的螢幕上，而捲動畫面所提供的性能亦可被用來做爲其它事。這種方法爲計算每一個位元組要印在螢幕上的螢幕記憶體位置，且POKE此位元組到此位置內。那種方式的不同變化如程式範例所示。在這個例子中，監督程式的VTAB常規被用來幫助建立一記憶體位置的表，以指出螢幕列的起始點。這是一種較比"第四頁到拾壹頁"單元中所描述的使用模組化算術表示，來得簡單的改變方式。注意加1ϕ24 到表中的每一個值，使此列的記憶位址是在次顯示區內。

位址表　ADDRESS TABLE

功　能	十六進位位　址	正十進位位　址	負十進位位　址	監督程式標　誌	破壞的暫存器
捲動畫面之外					
計算A暫存器所指列的記憶體位址；設定GBASL,H。	F847	63559	-1977	GBASCALC	A
捲動畫面之內					
使用INVFLG將位於A暫存器內的位元組寫到螢幕上，游標(CV),(CH)處，並提供游標移動。	FDFφ	65φφ8	-528	COUT1	?A
有游標移動但不用INVFLG而將位於A暫存器內的位元組寫到螢幕上(CV),(CH)位置。	FDF6	65φ14	-522	COUTZ	none
把捲動畫面清爲空白，游標移至左上角。	FC58	646φφ	-936	HOME	A,Y
用A暫存器設定CV，清除畫面至畫面的結尾處。	FC5A	646φ2	-934		A,Y

功能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
將游標放在列(A)行(CH)處，用A暫存器設定CV和BASL,H。	FB5B	64347	-1189	TABV	A
以CV(和WND-LFT)設定BASL,H。	FC22	64546	-99φ	VTAB	A
以(A)和WNDLFT設定BASL,H而不參看CV。	FC24	64548	-988	VTABZ	A
設定BASL,H爲A暫存器所指螢幕列(非畫面列)的左端。	FBC1	64449	-1φ87	BASCALC	A

CH	36	$24	WNDLFT	32	$2φ
CV	37	$25	WNDWDTH	33	$21
GBASL,H	38-39	$26-27	WNDTOP	34	$22
BASL,H	4φ-41	$28-29	WNDBTM	35	$33
INVFLG	5φ	$32			

Applesoft程式範例

APPLESOFT SAMPLE PROGRAM

```
10      REM  TEXT OUTPUT WITHOUT THE SCROLL WINDOW
11      REM  SAMPLE PROGRAM
12      REM  READS FROM KEYBOARD  LINE, CHAR, STRING
14      REM  AND PLACES THE STRING THERE
1000    REM  PROGRAM ENTRY
1010    DIM L%(23):      REM  LINE ADDR TABLE
1100    GOSUB 63000:     REM  MAKE UP TABLE
1199                     REM  PRINT PART OF TABLE JUST FOR SHOW
1200    FOR I = 0 TO 21: PRINT I,L%(I): NEXT
1209                     REM  DELAY TO ALLOW LOOK AT IT.
1210    FOR I = 1 TO 5000: NEXT
1220    PRINT:     REM  PRINT CR TO ALLOW CTL-S STOP IF DESIRED.
1225    CALL  - 936:     REM CLEAR SCREEN BEFORE CHANGING WINDOW.
1229                     REM  SET UP NEW WINDOW.
1230    POKE 32,24: POKE 33,14: POKE 34,12: POKE 35,17
1235    CALL -936:       REM  PUT CURSOR INTO WINDOW AREA.
1300    INPUT LI,CL,SS$:REM  READ A COMMAND LINE.
1399                     REM  ALLOW A WAY OUT
1400    IF SS$ = "END" THEN 63900
1500    SL =  LEN (SS$)
1509                     REM CHECK LEGALITY OF LINE, ETC.
1510    IF LI > 23 THEN 1810
1511    IF CL > 39 THEN 1810
1519                     REM NOT PAST 40 THOUGH.
1520    IF CL + SL > 39 THEN SL = 40 - CL
1600                     REM  PUT CHARACTERS ONE AT A TIME.
1601    FOR I = 1 TO SL
1700    C$ =  MID$ (SS$,I,1):C% =  ASC (C$)
1720    POKE L%(LI) + CL + I - 1,C% + 128
1740    NEXT I
1800    GOTO 1300:       REM GO BACK FOR ANOTHER COMMAND.
1810                     REM  LINE OR CH TOO BIG - ERROR.
1811    CALL  - 936: PRINT "NOT SO BIG"
1812    PRINT "LN ";LI: PRINT "CH ";CL
1820    GOTO 1800
62999                    REM
63000                    REM  MAKE UP LINE ADDRESS TABLE
63010   X% =  PEEK (37): REM  REMEMBER CV
63020   FOR I = 0 TO 23
63030   POKE 37,I:       REM  SET CV
63031   CALL -990:       REM  CALL VTAB TO FILL BASL & BASH
63035   L%(I) = 256 * ( PEEK (41)) +  PEEK (40)
63040   NEXT I
63045                    REM  TABLE SETUP DONE
63046                    REM  RESTORE CV AND RETURN
63050   POKE 37,X%: CALL  - 990: REM  WITH PROPER BASL & BASH
63060   RETURN
63900   CALL  - 1233: END: REM RESTORE FULL WINDOW PRIMARY
```

次顯示區域
SECONDARY DISPLAY AREAS

Apple II的硬體允許使用兩個記憶體區域的任一個爲螢幕顯示區。第一（或者主要的）爲記憶體位置 $φ4φφ-$φ7FF 。次要的文字（和低解析度圖形）顯示區域爲 $φ8φφ-$φBFF。這個區域通常爲使用者程式或資料覆蓋，但是在特殊情況下，使用者可能希望將此次要區域當做一個螢幕顯示區域。

監督程式並不太支援此次顯示區域。那就是，在監督程式中，由列號碼（CV）和字行（character column）（CH）以決定出螢幕區域記憶體位址的常規，僅對主要顯示區域做如此的動作。這些常規僅能夠在在列φ-23正常工作。

下面是使用次顯示區域的兩種方法之說明。

將主要抄至次要顯示區
COPY PRIMARY TO SECONDARY

時常我們希望很快地改變顯示區域（雖然程式製造的輸出非常慢）。舉例而言，一個程式可能要顯示由掃瞄磁碟檔而找出的資料，程式設計者可能在主要顯示區域內產生原來的螢幕資料，然後把它移至次顯示區域並設定硬體來顯示次要的記憶體。當操作者在觀看起始或者原先顯示的結果時，此程式繼續在主要區域內產生下一個螢幕資料

。在本節後段中，將舉一個例子程式，顯示監督程式的搬移常規如何把主顯示區的內容搬至次顯示區。

爲次顯示頁設定BASL, H
SET BASL,H FOR SECONDARY DISPLAY PAGE

當監督程式將一個字元放入螢幕記憶區域內時，它利用 BASL , H 爲列左端的記憶位置之位址，而（CH）爲此列由左端開始的位移（displacement）。BASL , H 可以經由設定所希的列號碼到 C V內，然後叫用 TABV，而被起始設定爲所指螢幕列的記憶體位置。當從那個CALL 返回時，加 4 到 BASH，把 BASL , H 改爲指向位於次顯示區域內所指列的記憶位置。這將持續直到此程式寫一個 carriage return 或者寫入的字元超過捲動畫面的右端爲止。

如果監督程式被叫用以便從鍵盤讀取資料，它"回映（echoes）"輸入的字到螢幕上。輸入一個 carriage return，一個 backspace，一個游標（cursor）的移動，或者一個清除螢幕的 Escape 鍵功能，將使監督程式將 BASL , H 重新取回以指到主要顯示區域內。

在一個顯示區域被用做文字顯示而另一個顯示區被用做圖形顯示的情況，讓圖形顯示在主要區域且文字在次要區域是比較好的，因爲監督程式在畫圖功能時，一直不斷地重新計算 GBASL , H；而對於文字輸出時，BASL , H 是僅在把游標移至新列時才重新計算。

此必須注意的是APPLESOFT亦沒有（很容易）支援此次顯示區域。RAM的APPLESOFT亦佔了這部份的記憶體。Firmwave的APPLESOFT除非採取特殊的動作，亦將程式碼擺在那塊記憶體空間。那些動作在舉例的程式中有註明，此程式使用APPLESOFT和次要的顯示區域。在放入程式之前，POKE 1φ4,12和3φ27,φ。

位址表 ADDRESS TABLF

功能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
將游標（cursor）放在列(A)，行（CH)(將A存放在CV且JMP到VTAB以計算BASL,H）。	FB5B	64347	-1189	TABV	A
經由叫用BAS-CALC和加上WNDLFT，用CV和WNDLFT值設定BASL,H。	FC22	64546	-99φ	VTAB	A
經由叫用BAS-CALC和加上WNDLFT 用A暫存器和WNDLFT來設定BASL,H。	FC24	64548	-988	VTABZ	A

功　能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
設定BASL,H爲A暫存器所指螢幕（不是畫面）列左端字元的記憶位址。	FBC1	64449	-1φ87	BASCALC	A
使用INVFLG之值將位於A暫存器內的位元組寫到螢幕上游標（CV),(CH)處，並提供游標移動。	FDFφ	65φφ8	-528	COUT1	?A
而將A暫存器內的位元組寫到螢幕上（CV),（CH）處，有游標移動但不用INVFLG。	FDF6	65φ14	-522	COUTZ	none
監督程式命令處理器MOVE常規。位置（A1L,H)到(A2L,H）被搬移至（A4L,H）位置起。	FE2C	65φ68	-468	MOVE	A(Y=φ)
監督程式命令處理器GO的進入點。 如果進入的話，用A1L,H	FEB6	652φ6	-33φ	GO	A,X,Y,P

功能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
處來設定PCL，H。					&
叫用RESTORE，設定除了S外的所有暫暫存器。	FEB9	652φ9	-327		&
根據PCL,H而JMP。	FEBC	65212	-324		

直接控制的位址 DIRECT CONTROL ADDRESSES

下表描述"經由直接控制（而不是叫用監督程式）來顯示到不同螢幕結構"的設定方法。這些項目之中，有些在監督程式內並無常規可被叫用以執行其功能。

功能（Function）	方法（Method）
設定GRAPHICS顯示模式。	POKE -163φ4,φ 或 STA Cφ5φ
設定文字顯示模式。	POKE -163φ3,φ 或 STA Cφ51
設定圖形模式爲完全螢幕。	POKE -163φ2,φ 或 STA Cφ52
設定混合圖形和文字模式。	POKE -163φ1,φ 或 STA Cφ53
設定主要頁爲顯示區。	POKE -163φφ,φ 或 STA Cφ54
設定次要頁爲顯示區。	POKE -16299,φ 或 STA Cφ55
清除HIRES=設定LORES 模式	POKE -16298,φ 或 STA Cφ56
設定HIRES 圖形模式。	POKE -16297,φ 或 STA Cφ57

功能（ Function ）	方法（ Method ）
設定捲動畫面的頂線。	POKE 34, 列－號碼（ ϕ-23 ） 底列必須大於頂列。
設定捲動畫面的左邊邊緣。	POKE 32, 行－號碼（ ϕ-39 ） 左邊邊緣＋寬度不能超過 4ϕ 。
設定捲動畫面的寬度。	POKE 33, 行數（ 1-4ϕ ）， 左邊邊緣＋寬度不超過 4ϕ 。
設定捲動畫面的底線。	POKE 35, 列－號碼（ 1-24 ） 底列必須大於頂列。

CH	36	$24	WNDLFT	32	2\phi$
CV	37	$25	WNDWDTH	33	$21
GBASL,H	38-39	$26-27	WNDTOP	34	$22
BASL,H	4ϕ-41	$28-29	WNDBTM	35	$23
INVFLG	5ϕ	$32			

整數BASIC程式範例
INTEGER BASIC SAMPLE PROGRAM

```
   10 REM  SAMPLE SECONDARY DISPLAY WAY
   11 REM  USING MONITOR MOVE TECHNIQUE

   19 GOTO 1000:                REM  BYPASS SUBROUTINES

   20                REM  MOVE AREA 1 TO AREA 2
   21 POKE 60,0: POKE 61,4:    REM SET A1L,H
   22 POKE 62,255: POKE 63,7: REM SET A2L,H
   23 POKE 66,0: POKE 67,8:    REM SET A4L,H
   25 POKE 71,0:               REM SET Y-REG=0
   26 POKE 58,44:              REM $2C
   27 POKE 59,254:             REM $FE
   28 CALL -327:               REM DO THE MOVE
   29 RETURN

 1000             REM  PROGRAM START
 1001 IF PEEK (75)<12 THEN 32000

 1100 CALL -936:              REM  CLEAR THE SCREEN

 1200 PRINT "THIS IS THE SECONDARY DISPLAY AREA"
 1210 PRINT "NOTE THE LACK OF CURSOR"

 1300 GOSUB 20:               REM  MOVE TO SECONDARY

 1400 CALL -936:              REM  CLEAR PRIMARY AGAIN
 1410 PRINT "THIS IS THE PRIMARY AREA AGAIN"
 1500 POKE -16299,0:          REM  SET SECONDARY
 1600 FOR I=1 TO 4000: NEXT I
 1700 POKE -16300,0:          REM  BACK TO PRIMARY
 1800 END

32000            REM  NO LOMEM ERROR
32001 PRINT "PLEASE LOAD AGAIN"
32002 PRINT "AFTER LOMEM:3072 "
32003 END
```

Applesoft程式範例
APPLESOFT SAMPLE PROGRAM

```
  10          REM SECONDARY DISPLAY AREA WAYS AND MEANS
  11          REM SAMPLE PROGRAM
  12          REM READS FROM KEYBOARD
  13          REM COMMAND, LINE, CHARACTER, STRING
  14          REM AND PLACES THE STRING

1000          REM PROGRAM ENTRY

1009          REM IS SECONDARY AREA CLEAR?
1010  IF  PEEK (104) < 12 THEN 62000

1020  GOSUB 63000:              REM CLEAR THE SECONDARY
1300  IF Q = 0 THEN 1390:       REM INPUT TO PRIMARY
1309                            REM SET INPUT TO SECONDARY
1310  POKE 37,21:               REM SET LINE 21
1311  CALL  - 990:              REM SET BASL,H
1312  POKE 41, PEEK (41) + 4:   REM SET BASH TO SECONDARY

1390  INPUT CC$,LI,CL,SS$
1400  IF CC$ = "END" THEN 63900
1410  IF CC$ = "S" THEN 2000:   REM SET SHOW TO SECONDARY AREA
1420  IF CC$ = "P" THEN 2100:   REM SET SHOW TO PRIMARY AREA
1430  IF CC$ = "Q" THEN 2200:   REM SET INPUT SECONDARY
1431  IF CC$ = "R" THEN 2300:   REM SET INPUT PRIMARY
1440  IF CC$ = "X" THEN 1500:   REM PUT STRING TO SECONDARY
1450  POKE 16300,0: PRINT "WHAT? ": GOTO 1300
1500  SL =  LEN (SS$)
1510  IF LI > 23 THEN 1810
1511  IF CL > 39 THEN 1810
1512  IF CL + SL > 39 THEN SL = 40 - CL: REM NO AUTO CR
1590  CX =  PEEK (37):          REM REMEMBER CV
1600  POKE 37,LI: CALL  - 990: POKE 41, PEEK (41) + 4
1610  POKE 37,CX:               REM RESTORE CV
1620  POKE 36,CL:               REM SET CH FOR THIS PRINT
1700  SP$ =  LEFT$ (SS$,SL):    REM SHORTEN PRINT IN THIS SMPL
1710  PRINT SP$
1800  GOTO 1300
1810  CALL  - 936:              REM VALUE TOO LARGE.
1811  PRINT "NOT SO BIG":       REM PRINT IN PRIMARY ONLY
1812  PRINT "LN ";LI: PRINT "CH ";CL
1820  GOTO 1300

2000  POKE  - 16299,0:          REM SET SECONDARY
2010  GOTO 1300

2100  POKE  - 16300,0:          REM SET PRIMARY
2110  GOTO 1300

2200  Q = 1: GOTO 1300:         REM SET INPUT TO SECONDARY
2300  Q = 0: GOTO 1300:         REM SET INPUT TO PRIMARY
```

```
62000  PRINT "SETUP NOT MADE, NOW BEING DONE"
62010  PRINT "RUN THE PROGRAM AGAIN"
62018            REM 104 IS APPLESOFT ROM START
62019         REM BYTE BEFORE $C01 MUST BE ZERO
62020  POKE 3072,0: POKE 104,12: END

63000  BL$ = "     ":            REM CLEAR SLCONDARY AREA
63001  FOR I = 1 TO 3:BL$ = BL$ + BL$: NEXT
63005  CX =  PEEK (37)
63010  FOR I = 0 TO 23
63020  POKE 37,I: CALL  - 990
63030  POKE 41, PEEK (41) + 4
63040  POKE 36,0
63050  PRINT BL$
63060  NEXT
63070  POKE 37,CX: POKE 36,0
63080  RETURN
63900  POKE 16300,0: CALL  - 1233: END
```

第三章
插斷處理
INTERRUPT PROCESSING

某些電腦是能夠對一信號線的上昇（或下降）有所反應，它的反應是瞬間地將處理器目前的狀況保留起來，且迅速地將控制權轉移給電腦內的某一其它程式。那條線狀態的改變稱之爲"造成一個插斷"。處理器保留其目前的狀態而將控制權轉移給記憶中的某些位置，這種功能稱之爲"接受一個插斷"。接到控制權的那程式便要"處理此插斷"。

在Apple II 中的65φ2 處理器對於三種插斷有反應。它們是RESET, NMI（Non-Maskable Interrupt），和IRQ。執行一個BRK指令，將會造成模擬一個IRQ形式的插斷。

一般而言，一個插斷的目的爲允許某些外面的裝置，把它的狀態通知正在跑的程式而不需此正在跑的程式週期性地或連續性地測試硬體的狀態。這種作業的例子如Apple II 的鍵盤作業。當鍵盤輸入要被接受時，記憶體位置 $Cφφφ 被反覆的測試，直到 sign 位元顯示一個鍵已經被按下去了。一個插斷驅動處理的例子如一特殊的週邊控制卡，接上電話線上，在任何時間資料可取用時，插斷電腦讓資料擷取程式拿取資料，但是在資料傳輸間隔可以允許機器做其它的事。

當一個電腦承認接受一個插斷時，其硬體應該完成三件事。

1. 保留處理器的狀態，因此當此插斷被“服務”或者處理完了以後，被插斷的程式能夠繼續執行。
2. 防止再度地接受同種形式的插斷，直到插斷處理程式重新恢復原來可以被插斷的樣子以後，才可以再度地接受。
3. 將控制權傳給某個“處理這種型式或種類的插斷”的程式。

在Apple II中所用的6502對上面三種步驟的變化，歸爲三種不同的插斷種類。

1. 當一個IRQ（或BRK）或NMI 插斷被發生時，程式計數器（program .counter）和此P暫存器（處理器的狀態暫存器）被分別壓入堆疊器（stack）上。當一個RESET 插斷發生時，處理器把記憶體維持爲READ 模式，直到控制權是被交處理者（handler），所以沒有任何處理器間的狀態被壓入堆疊器（stack）上。

2. 當 65ϕ2 發生一個IRQ 插斷時，P 暫存器被修改。如果執行一個BRK 指令，在P暫存器被壓入堆疊器（stack ）之前，則處理器狀態暫存器的1\phi$ 位元被設定爲1。如果是 IRQ 線造成此插斷，則在P暫存器被壓入堆疊器之前，此位元被設定爲零。

在P暫存器被壓入堆疊器之後，P暫存器的$04位元被設定爲禁止再接受 IRQ 種類的插斷，直到此插斷處理程式清除掉此狀況爲止。

對RESET 和NMI ，當目前的插斷正被處理時，並沒有可用的方法以防止其它的插斷。

3. 此 65φ2 經由“向量 vectors ”方式而將控制權交給適當的程式以便處理插斷。記憶體位址 $FFFA-$FFFF爲保留做此目的。發生插斷的最後一個步驟是把此類插斷的向量載入程式計數器（program counter）中，以開始執行插斷處理程式。下面的表爲兩種監督程式，插斷處理程式的位置。

插斷的發　生	向量位址	監督程式的標誌	舊型監督程式位址	自動開始監督程式的位址
NMI	$FFFA-B	“NMI”	$φ3FB	$φ3FB
RESET	$FFFC-D	RESET	$FF59	$FA62
IRO/BRK	$FFFE-F	IRQ	$FA86	$FA4φ

NMI插斷　NMI INTERRUPT

Apple II 的監督程式並不干擾使用者處理NMI 的插斷。亦即，NMI 的向量使 65φ2 將電腦的控制權交給 $φ3FB 位置，在這裡使用者可以擺放一個 JMP ，以跳至使用者所提供的處理這種插斷的處理程式。

RESET插斷　RESET INTERRUPT SUPPORT

按下了鍵盤上的RESET鍵，將造成一個RESET 插斷的發生，除了非常早期的Apple II 外，所有的Apple II 在電源打開時，亦會產

生一個RESET插斷。

自動開始的監督程式和舊型的監督程式，兩種RESET 插斷處理程式所執行的動作，有相當大的不同。因此，它們將被分開地描述。

IRQ/BRK插斷處理
IRQ/BRK INTERRUPT HANDLING

當一個 IRQ 插斷被發生或者一個BRK 指令被執行，則此 65ϕ2執行一插斷程序。程式計數器的內容被壓入堆疊器。P 暫存器的 $1 ϕ位元被設定或者清除以指出是 IRQ 線或 BRK 指令插斷，然後它壓入堆疊器。然後 65ϕ2 把P暫存器的 $\$\phi$4 位元設爲 1 ，以防止這種型式的其它插斷再度地被接受，直到這個插斷被處理完以後爲止。然後6502把IRQ硬體事先定好，位於 $FFFE-$FFFF 內的向量載入程式計數器(program counter)，且允許電腦的操作從那一點開始。IRQ中斷的插斷處理程式現在已經接掌了控制權。

RESET插斷—舊型的監督程式
RESET INTERRUPT—OLD MONITOR

當一個RESET 插斷發生時，則舊型的監督程式建立一個預先設定好的硬體架構和第零頁的欄位。主要地，鍵盤被設定爲目前的輸入裝置，而螢幕被設定爲目前的輸出裝置，且此螢幕的結構被設定爲完全螢幕，且爲一般視頻的捲動畫面。

第零頁欄位KSWL,H,CSWL,H 被設定以使鍵盤和螢幕爲有效狀態（Active）。WNDLFT, WNDWDTH, WNDTOP, WNDBTM 被設定以定義此整個螢幕爲捲動畫面。CV 和CH 被設定爲將游標（cursor）放在螢幕的左下角。INVFLG 被設定爲正常模式（黑底白字）。

這些硬體位址被參考以建立一已知的架構如下：

C\phi$56 —— 清除高解析度的圖形。

C\phi$54 —— 顯示主要顯示區。

C\phi$51——設設定文字模式。

控制權因此被交給到監督程式的"開頭"，標識爲MON處，其位置爲$FF65，此處鈴聲會響且監督程式進入"命令列讀取常規"。

位址表　ADDRESS TABLE

功　能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
把位於SAVE區域內的STATUS設爲ϕ。	& FB2F	643ϕ3	-1233	INIT	A
清除HIRES。	& FB33	643ϕ7	-1229		A
設定主要顯示區域。	& FB36	6431ϕ	-1226		A
設定文字模式。	& FB39	64313	-1223	SETTXT	A
把(A)=ϕ而跳至SETWND 以設定爲全螢幕的捲動畫面。	FB3C	64316	-122ϕ		A

功能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
用A暫存器設定&WNDTOP。	FB4B	64331	-12ϕ5	SETWND	A
放ϕ到A暫存器&內以便給WNDLFT。	FB4D	64333	-12ϕ3		A
用A暫存器設定&WNDLFT。	FB4F	64335	-12ϕ1		A
放4ϕ到A暫存器內以便給WNDWDTH。	FB51	64337	-1199		A
用A暫存器設定&WNDWDTH。	FB53	64339	-1197		A
放24到A暫存器&內以便給WNDBTM。	FB55	64341	-1195		A
用A暫存器設定&WNDBTM。	FB57	64343	-1193		A
放23到A暫存器&內以便給CV。	FB59	64345	-1191		A
用A暫存器設定&CV。	FB5B	64347	-1189	TABV	A
跳至VTAB以設定BASL,H並RTS。	FB5D	64349	-1187		A
設定INVFLG爲$FF=正常的視頻。	FE84	65156	-38ϕ	SETNORM	A
用Y暫存器設定INVFLG。	FE86	65158	-378	SETIFLG	none
設定埠ϕ(鍵盤	FE89	65161	-375	SETKBD	A,X,Y

功　能	十六進位位　址	正十進位位　址	負十進位位　址	監督程式標　誌	破壞的暫存器
)爲輸入。					
設定埠φ(螢幕)爲輸出。	FE93	65171	-365	SETVID	A,X,Y
當RESET 鍵壓下去或打開電源時監督程式的進入點。	FF59	65369	-167	RESET	
叫用SET-NORM——黑底白字。	&				
叫用INIT—文字&完全捲動。	& FF5C	65372	-164		
叫用SETVID——螢幕爲輸出。	& FF5F	65375	-161		
叫用SETKBD——鍵盤=輸入。	& FF62	65378	-158		
清除65φ2 十進位模式(設定爲16進位)。	& FF65	65381	-155	MON	
響鈴聲。	& FF66	65382	-154		
監督程式的命令處理器進入點。設定"*"爲提醒字元(prompt character)。	FF69	65385	-151	MONZ	

RESET插斷—自動開始的監督程式
RESET INTERRUPT—AUTOSTART MONITOR

自動開始的監督程式在處理一個RESET 插斷時，執行三種功能。

1. 根據基本的機器結構，建立一已知的硬體／軟體環境。

2. 如果記憶體的內容（第三頁）沒有指出開電源時的起動動作已被執行，則此自動開始的監督程式將執行此開電源時的起動動作。如果一個磁碟控制卡出現在其中一個槽，開電源的起動動作則包括從那個槽產生靴帶作用（ bootstrapping ）。如果機器上沒有磁碟控制卡，則模擬一個控制-B的輸入被執行。 任一個情況，則適當的語言處理器在開電源的起動動作結束時將接到控制權，設定第三頁的欄位來指出已是暖機開始（warm start），以保證再來RESET鍵會產生一個暖機開始的插斷。

3. 如果記憶體的內容（第三頁）指出開電源時的起動動作已經被執行過，自動開始的監督程式將在"處理" RESET 插斷的結尾處，透過位於第三頁的RESET （軟性進入 Soft Entry ）向量，而交出控制權。如果 DOS 已經被讀入，將會經由 DOS 而將控制權傳回到目前的語言處理器。如果 DOS 不存在，則正常設定的RESET 向量將造成一個控制C的模擬執行（暖機開始）再進入目前的語言處理器。

設定系統最初的架構
INITIALIZE SYSTEM CONFIGURATION

當一個RESET 鍵發生時，自動開始的監督程式建立一預先定義好的硬體架構和第零頁欄位。主要的是，此鍵盤被設定爲目前的輸入裝置，螢幕被設定爲目前的輸出裝置，且此螢幕結構被設定爲正常視頻的全螢幕捲動畫面。

第零頁中的欄位KSWL,H , CSWL,H被設定使鍵盤和螢幕爲可作用。WNDLFT , WNDWDTH , WNDTOP , WNDBTM被設定以把整個螢幕定義爲捲動畫面。CV 和CH 被設定以便把游標（ cursor)放在螢幕的左下角。 INVFLG 被設定爲正常模式（ 黑底白字 ）。

如下的硬體位址被參用以建立一已知架構。

$C\phi56$　　清除高解析度圖形。

$C\phi54$　　顯示主要區域。

$C\phi51$　　設定文字模式。

$C\phi58$　　清除ANϕ = TTL LO。

$C\phi5A$　　清除AN1 = TTL LO。

$C\phi5D$　　設定AN2 = TTL HI 。

$C\phi5F$　　設定AN3 = TTL HI 。

$CFFF　　關掉擴展的ROM。

$C\phi1\phi$　　清除鍵盤觸發信號（ keyboard strobe ）。

當完成上面所列時，此自動開始監督程式將會響鈴聲。

冷／暖機的決定 COLD/WARM DETERMINATION

當建立一已知的基本硬體和軟體（螢幕控制 screen control）的環境後，自動開始監督程式執行一個測試動作以判定開電源時的起動動作是否要被執行。第三頁位置 $φ3F2 — $φ3F3 含著RESET（軟性入口 soft entry）向量，自動開始的監督程式在完成處理此RESET插斷時，會將控制權傳給這向量之位址。位置 φ3F4 是一個有用的位元組，與 $φ3F3一起使用以決定是否要執行開電源的起動動作。如果這兩個記憶體位置的內容作Exclusive OR的結果爲 $A5，則認定開電源的起動動作在前面已經執行過了，且 $φ3F2-$φ3F3 被認爲是一有效的位址，以便用來傳遞控制權。

開電源時的起動 POWER-ON INITIALIZATION

開電源起動的第一個功能爲在第三頁（$φ3Fφ-$φ3F4）中建立BRK的插斷向量（看“BRK 指令的處理——自動開始的監督程式”）且在RESET 軟性入口插斷向量放入有效的位元組。在這點的RESET向量被設定爲 $Eφφφ 以便針對目前的語言處理器模擬一個控制—B（起動 initialize）。

自動開始監督程式接下來便執行一個常規——測試每一個槽（slot），從槽 7 到槽 1，以便看看磁碟控制器卡是否存在。如果有一個被發現了，則執行一個 JMP 到 φCXφφ，其中 X 爲磁碟控制器被發

現時的槽號碼。其結果是載入 DOS 且執行其HELLO程式。

注意：DOS 3.2 放回位於 $φ3F2-$φ3F3 的RESET 向量和位於 $φ3F4 的有效位元組，所以在一個RESET中斷時，控制權將經由DOS 交回到目前的語言處理器。

如果沒有發現磁碟控制卡，自動開始監督程式將RESET 向量改爲 $Eφφ3(語言重新開始或控制－C的進入點) 且跳至 $Eφφφ(語言起動的進入點)。

系統的重新開始　SYSTEM RESTART

如果上面所描述的 $φ3F3-$φ3F4 測試通過了，則位於 $φ3F2-$φ3F3 的RESET向量被認爲是最有效的。如果它含著 $Eφφφ，則它被改成爲 $Eφφ3 且由 $Eφφφ 進入 BASIC。如果它並不是 $Eφφφ，它透過 $φ3F2-$φ3F3 執行一個間接跳越 (Indirect Jump) 而跳至 ($φ3F2-$φ3F3) 其中所指的位址。

由使用者修飾的RESET向量
RESET VECTOR MODIFICATION BY USER

RESET 向量可能被使用者或程式修改，以便在監督程式完成插斷的處理之後，將控制權交給機器中的其它位址。舉例而言，要使按 RESET 鍵的結果是將機器置於監督程式模式，則執行下面的程式；

```
10 POKE 1010,105
20 POKE 1011,255
30 POKE 1012,90
40 CALL -151: REM ENTER MONITOR
50 END
```

下面的程式更爲常見。爲了要把RESET 向量設定爲某個位址，把這位址POKE到位置1φ1φ-1φ11（$φ3F2-$φ3F3）然後叫用自動開始監督程式中標誌爲SETPWRC（$FB6F或64367或－1169）的常規以設定位置1φ12（$φ3F4）之值。

```
10 REM AD IS ADDRESS OF
11 REM ROUTINE TO RECEIVE
12 REM CONTROL AFTER RESET
20 POKE 1010,AD: REM SET LO BYTE
30 POKE 1011,AD/256: REM SET HI
40 CALL -1169: REM SET 1012
```

注意：如果你試著在一個用舊型監督程式 ROM 的系統中執行這個程式，你可能會破壞此程式或者甚至整個磁碟。爲了避免這個問題在一個具有自動開始 ROM 的系統，單步地執行上面程式。然後，PEEK位置1φ12而得到此值以便POKE到舊型機器的1φ12 位置，而不必CALL-1169。

位址表 ADDRESS TABLE

功能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
當按下RESET鍵或開電源時，監督程式的	FA62	64φ98	-1438	RESET	

功　能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
進入點。					
CLD－清除 65φ2 的十進位，（設定爲16進位）。	&				
Call SET-NORM－黑底白字。	& FA63	64φ99	-1437		
Call INIT－文字完全螢幕。	& FA66	641φ2	-1434		
Call SETV-ID－螢幕當做輸出。	& FA69	641φ5	-1431		
Call SETK-BD－鍵盤當做輸入。	& FA6C	641φ8	-1428		
設定硬體爲一已知狀態。	FA6F	64111	-1425	INITAN	
清除ANφ爲 TTL LO（參考Cφ58）。	&				
清除AN1爲 TTL LO（參考Cφ5A）。	& FA72	64114	-1422		
設定AN2爲 TTL HI（參考Cφ5D）。	& FA75	64117	-1419		
設定AN3爲 TTL HI（參考Cφ5F）。	& FA78	6412φ	-1416		

功能	十六進位位址	正十進位位址	負十進位位址	監督程式破壞的標誌暫存器
清除擴大的ROM(參考CFFF)。	& FA7B	64123	-1413	
清除鍵盤的觸發信號。	& FA7E	64126	-141φ	
清除65φ2 十進位模式(設定爲十六進位)。	& FA81	64129	-14φ7	NEWMON
叫用BELL。	& FA82	6413φ	-14φ6	
測試$3F3與$3F4：冷機或暖機。 如果冷機跑至PWRUP。 如果($3F3)XOR($3F4)=$A5，則暖機。	FA85	64133	-14φ3	
測試SOFTEV($3F2)之低位元組： 不爲零表示做冷機開始—跑至NOFIX以使用SOFTEV向量。 零表示可能爲重新開始的暖機。	FA8F	64143	-1393	
測試SOFTEV	FA94	64148	-1388	

功　能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
的高位元組是否爲 Eφ——是則至語言的冷機開始進入點。如果不等的話，SOFTEV可以被使用，跑至NOFIX。					
此時SOFTEV＝Eφφφ，把它改爲Eφφ3以備未來使用且跑至Eφφφ以便冷機開始此語言。	FA9B	64155	-1381	FIXSEV	
JMP(SOFTEV)：使用軟性入口向量(Soft Entry vector)以離開RESET的處理程式。	FAA3	64163	-1373	NOFIX	
在RESET進入點冷機開始。叫用APPLE Ⅱ以清除螢幕且將標題(title)放在頂列上。　&	FAA6	64166	-137φ	PWRUP	

功能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
設定第三頁的插斷向量以便給BRK（OLD-BRK）和SOFTEV（$Eϕϕϕ）用。	FAA9	64169	-1367	SETPG3	
在槽7到槽1尋找磁碟控制卡。如果沒有的話，則跑至前面的FIXSEV來設定SOFT-EV以便BASIC的重新開始&進入BASIC的冷機（cold）。 如果發現磁碟JMP（LOCϕ）以便從磁碟讀入DOS。	FAB4	6418ϕ	-1356		
清除螢幕（叫用&HOME）。	FB6ϕ	64352	-1184	APPLEⅡ	A,Y
將APPLEⅡ標示置於頂列上。	FB63	64355	-1181		A,Y
設定PWREDUP（$3F4）=（$3F3）XOR $5A	FB6F	64367	-1169	SETPWRC	A
把位於SAVE區&	FB2F	643ϕ3	-1233	INIT	A

功　能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
域內的STATUS設爲φ。&					
清除HIRES。&	FB33	643φ7	-1229		A
設定主要的顯示&區域。	FB36	6431φ	-1226		A
設定文字模式。&	FB39	64313	-1223	SETTXT	A
定(A)＝φ且跳至SETWND 以設定完全螢幕的捲動畫面(full screen scroll window)。	FB3C	64316	-122φ		A
用A暫存器設定&WNDTOP。	FB4B	64331	-12φ5	SETWND	A
把φ放入A以便&給WNDLFT用。	FB4D	64333	-12φ3		A
以A暫存器來設&定WNDLFT。	FB4F	64335	-12φ1		A
將4φ放入A以&便給WNDWDTH用。	FB51	64337	-1199		A
以A暫存器來設&定WNDWDTH。	FB53	64339	-1197		A
將24放入A以便&給WNDBTM用。	FB55	64341	-1195		A
以A暫存器來設&定WNDBTM。	FB57	64343	-1193		A

功能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
將23放入A以便給CV用。	FB59	64345	-1191		A
以A暫存器來設定CV。	FB5B	64347	-1189	TABV	A
跳至VTAB以設定BASL,H&RTS。	FB5D	64349	-1187		A
設定INVFLG爲$FF=正常視頻。	FE84	65156	-38ϕ	SETNORM	Y
以Y暫存器來設定INVFLG。	FE86	65158	-378	SETIFLG	none
設定埠ϕ(鍵盤)爲輸入。	FE89	65161	-375	SETKBD	A,X,Y
設定埠ϕ(螢幕)爲輸出。	FE93	65171	-365	SETVID	A,X,Y
爲了要與舊型的監督程式能夠互相配合，此RESET 常規仍舊在這裡。叫用SET-NORM 黑底白字上。	FF59	65369	-167	OLDRST	
叫用INIT &文字&全螢幕	FF5C	65372	-164		
叫用SETVID &一螢幕爲輸出。	FF5F	65375	-161		
叫用SETKBD &	FF62	65378	-158		

功　能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞暫存器
一鍵盤＝輸入。					
清除65φ2 的十進位模式，設定爲16進位。	& FF65	65381	-155	MON	
響鈴聲。	& FF66	65382	-154		
監督程式命令處理器的進入處。設定"＊"爲提醒字元（prompt）。	FF69	65385	-151	MONZ	

IRQ/BRK插斷
IRQ/BRK INTERRUPTS

IRQ/BRK插斷的認可
IRQ/BRK INTERRUPT RECOGNITION

當一個 IRQ 插斷發生或執行一個BRK指令時，65φ2執行一個插斷順序。程式計數器（program counter）的內容被壓入堆疊器（stack）。P暫存器的$1φ位元被設定或清除，以指出是 IRQ 線或BRK指令產生插斷，且此暫存器被壓入堆疊器（stack）。65φ2 則設定P暫存器的$φ4位元，以防止這種型式的其它插斷再度被認可，而直到這個插斷被處理完爲止。65φ2把IRQ硬體所規定的"位於$FFFE-$FFFF 的向量"載入程式計數器（Program Counter）內

，且允許電腦從那一點繼續運轉。此 IRQ 插斷的插斷處理器現在接管了控制權。

IRQ插斷處理　IRQ INTERRUPT HANDLING

在舊型的監督程式和自動開始的監督程式中，65φ2位於 $FFFE-$FFFF 的指引向量，均指向監督程式中標誌（ label ）爲 IRQ處。在位址表中有註明其位址是不同的。

在兩種監督程式中，對於 IRQ插斷的處理是一致的。A暫存器的內容是被貯存在ACC（$45） 以供未來參考之用。由於發生插斷而被壓入堆疊器（ stack)的處理器狀態（ P暫存器），現被拿出到A暫存器內，然後又壓入堆疊器內，所以堆疊器和其指標（ pointer)都沒有改變。憑著把A暫存器向左移動三個位元，IRQ常規把它移入正負號位元（ sign bit ）的那個位元（ 在此例子中爲－0 ）指出此插斷爲一IRQ插斷而不是執行一個 BRK 指令。這時監督程式透過位置$φ3FE-$φ3FF執行一個間接跳越指令，以跳至使用者所提供的 IRQ插斷處理程式。注意在一個 IRQ 插斷，X, Y 和 S 暫存器沒有被監督程式保留。而且，插斷處理程式在跳出時有責任清除掉 $φ4 位元讓未來的插斷能發生。

BRK指令插斷　BRK INSTRUCTION INTERRUPT

執行一個 BRK 指令，造成 65φ2 模擬一個 IRQ 插斷而僅有少許的不同。由於 BRK 指令執行，壓入堆疊器的位址爲插斷模擬之一部份）是 BRK 指令之後兩位元組。

在把P暫存器壓入堆疊器之前，$1φ位元被設定以告訴插斷處理常規造成此插斷的原因是執行一BRK指令而不是IRQ線。在將P暫存器壓入堆疊器後，$φ4位元被設定以防止IRQ插斷再被認可，直到插斷處理程式清除掉此情況。然後控制權根據65φ2插斷向量交給監督程式標誌爲IRQ處。如同上面所描述關於IRQ插斷的處理，此IRQ常規首先貯存A暫存器在ACC($45)處以供未來參考，然後使用A暫存器去測試被堆疊的P暫存器內容，看在$1φ位元是否爲1。堆疊器和堆疊器指標(stack pointer)在此動作並沒有被改變。這個測試的結果爲將控制權傳給監督程式標誌BREAK處。注意在位址表中兩種監督程式的BREAK位址並不相同。

BRK指令—狀態的保留
BRK INSTRUCTION—SAVING OF STATUS

在每一種監督程式裡，其BREAK常規所做的第一件事爲保留完整的機器狀態在第零頁內。A暫存器的內容在進入IRQ插斷處理程式之前被貯存起來。BREAK常規從堆疊器中將被堆疊的P暫存器內容拿出來，且做一個JSR到SAV1，在此點其餘的暫存器被保留起來。注意這會清除P的$φ4位元，允許未來的IRQ或BRK插斷會發生。無論如何，在那個時間被保留的S暫存器由於將P暫存器從堆疊器拿回(POP)，而已經加1一次，且由於JSR到SAV1而減1兩次。當從SAV1返回，BREAK常規從堆疊器中拿回程式計數器且將它貯存在第零頁的位置PCL-PCH。在本節結束處的位址表指出在第零頁中用來貯存上面項目的位置。

BRK指令—舊型的監督程式
BRK INSTRUCTION—OLD MONITOR

舊型監督程式的 BRK 指令插斷處理程式的功能爲經由COUT顯示碰到 BRK 指令時的機器狀態，然後將控制權交回到監督程式頂端標誌爲 MON 處。上面的細節爲描述經由貯存機器狀態在第零頁中，包括PCL,H。舊型監督程式BREAK常規接下來做一個 JSR到 INSDS1 ，以顯示由 PCL-PCH 所指位址的指令（此爲在被執行BRK指令的下兩個位元組），並 JSR 到RGDSP1以顯示五個暫存器的內容，P,A,X,Y,S 。注意S 暫存器所顯示的值是比“ BRK 執行至 JSR至 SAV1 期間的值”小 2 。當完成暫存器的顯示時，一個JMP到MON以結束此插斷的處理。

BRK指令—自動開始的監督程式
BRK INSTRUCTION—AUTOSTART MONITOR

自動開始監督程式處理 IRQ 插斷（一個 BRK 指令插斷）是把暫存器和程式計數器貯存在第零頁位置。自動開始監督程式的BREAK常規，然後透過位於 $φ3Fφ-$φ3F1 的Apple-II BREAK向量而離開。所以，一個使用者程式在那一點得到控制權且做些顯示暫存器以外的動作並返回到監督程式的命令處理器，這是可能的。如此的一個程式必須保證在返回時，清除掉位於P 暫存器內的 $φ4 位元。在對開電源時的RESET 插斷處理時，這個向量被設定初值以指向自動開始監督程式中，標誌爲OLDBRK 處，這個常規與舊型監督程式所做的事相同。亦即，它做一個 JSR 到 INSDS1 以顯示由 PCL-PCH 所指位

置的反組合指令，做一個JSR到RGDSP1 以顯示暫存器之內容，然後一個JMP到MON以完成插斷的處理。

注意：DOS 3.2 於 bootstrap 動作時，會破壞第三頁，它重新存放這個向量以指向 $FA59 , OLDBRK 。

位址表　ADDRESS TABLE

功　能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
將位於（PCL,H）的指令反組合，而經由COUT印出。	F8Dφ	63696	-184φ	INSTDSP	A,X,Y
在carriage return後，從保留區域，經由COUT以顯示暫存器的值。	FAD7	64215	-1321	REGDSP	A,X
從保留區域，經由COUT 來顯示暫存器。	FADA	64218	-1318	RGDSP1	A,X
保留65φ2 的暫存器於 $45-49 。	FF4A	65354	-182	SAVE	A,X
保留A暫存器&在ACC $45。					
保留X暫存器&在XREG $46。	FF4C	65356	-18φ	SAV1	
保留Y暫存器&在YREG $47。	FF4E	65358	-178		

功能		十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
保留P暫存器在STATUS $48。	&	FF5φ	6536φ	-176		
保留S暫存器在SPNT $49。	&	FF54	65364	-172		
清除65φ2 十進位模式（設定16進位）。						
清除65φ2 十進位模式（設定16進位）。	&	FF65	65381	-155	MON	
響鈴聲。	&	FF66	65382	-154		
監督程式命令處理器的進入處。設定"*"爲提醒字元。		FF69	65385	-151	MONZ	
自動開始IRQ/BRK的處理。						
判定插斷爲IRQ或BRK，然後依此傳遞控制權。		FA4φ	64φ64	-1472	IRQ	A
處理BRK插斷：從堆疊器中從新取回P暫存器。		FA4C	64φ76	-146φ	BREAK	A,X,Y
保留暫存器（						

功　能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
SAV1)X, P,S。 將插斷位置從 堆疊器內移 到PCL,H。 JMP(BRKV) 到可能爲使 用者指定的 程式(一般 爲到OLDBRK ，如下面)。					
內是的 BRK 插 斷處理程式。 結束常規 顯示指令(下 2個位元組)。 顯示暫存器， JMP到 MON。	FA59	64φ89	-1447	OLDBRK	A,X,Y
舊型監督程式 IRQ/BRK 處理					
決定插斷是否爲 IRQ或BRK, 據此傳遞控制 權。	FA86	64134	-14φ2	IRQ	A
處理BRK插斷： 保留暫存器，	FA92	64146	-139φ	BREAK	A,X,Y

功能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
顯示指令（下兩個位元組），顯示暫存器，JMP 到MON					
PCL,H	58, 59	$3A, 3B	YREG	71	$47
ACC	69	$45	YSAV	52	$34
XREG	70	$46	STATUS	72	$48

第四章
雜碎整理 MISCELLANY

幫助發展機械語言
MACHINE LANGUAGE DEVELOPMENT AIDS

在監督程式內有很多常規在幫您發展機械語言程式時，是相當有助益的。其中有些常規可在已完成的程式使用，如監督程式的MOVE程式。表中的其他項爲一般，特殊，或非常特殊的螢幕輸出常規，和一些資料處理常規。

位址表

功　能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
將A暫存器內的位元組寫到螢幕上CV,CH處。	FDED	65φφ5	-531	COUT	?A
經由COUT印出carriage return。	FD8E	6491φ	-626	CROUT	A
經由COUT印出三個空白。	F948	63816	-172φ	PRBLNK	A,X
經由COUT印出(X)空白。	F94A	63818	-1718	PRBL2	A,X
印出A內的字元	F94C	63820	-1716	PRBL3	A,X

功能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
跟著印出(X)一個空白。					
經由COUT印出Bell碼。	FF3A	65338	-198	BELL	A
經由COUT印出"ERR"和BELL。	FF2D	65325	-211	PRERR	A
將A內的低4個位元以16進位字元印出。	FDE3	64995	-541	PRHEX	A
將A-暫存器的內容以兩個十六進位的4位元(nibble)印出。	FDDA	64986	-55φ	PRBYTE	A
以16進位印出Y,X暫存器。	F94φ	638φ8	-1728	PRNTYX	A
以16進位印出A,X暫存器。	F941	638φ9	-1727	PRNTAX	A
以16進位印出X一暫存器。	F944	63812	-1724	PRNTX	A
印出CR,然後再以16進位來印X,Y暫存器,再印個負號(或者dash)。	FD96	64918	-618	PRYX2	A,Y
以16進位印出Y,	FD99	64921	-615		A,Y

功　能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
X暫存器，然後dash。					
印出CR，16進位的A1H，A1L，和dash。	FD92	64914	-622	PRA1	A,X,Y
以16進位印出記憶體，內容前有其位址，從mmmm到mmm7，其中mmmm爲A1L,H 的起始值。	FDA3	64931	-6ϕ5	XAM8	A(Y=ϕ)
以16進位印出記憶體內容，範圍從（A1L,H）到（A2L,H）。	FDB3	64947	-589	XAM	A(Y=ϕ)
保留A,X,Y,P,S暫存器在$45-49。	FF4A	65354	-182	SAVE	A,X
前面先一個carriage return，而從$45-49 將所貯存的暫存器加上名稱顯示出來。	FAD7	64215	-1321	REGDSP	A,X
如同上面的方式	FADA	64218	-1318	RGDSP1	A,X

功能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
顯示暫存器而沒有CR。					
從$45處重新取回A,X,Y,P而沒有S。	FF3F	65343	-193	RESTORE	A,X,Y,P
監督程式的命令處理器GO的進入點。如果進入的話以A1L,H 設定PCL,H。 &	FEB6	652φ6	-33φ	GO	A,X,Y,P
叫用RESTORE,設定所有暫存器S除外。 &	FEB9	652φ9	-327		
透過PCL,H作跳越。	FEBC	65212	-324		
將記憶體內容從(A1L,H)到(A2L,H),移至以(A4L,H)開始。	FE2C	65φ68	-468	MOVE	A(Y=φ)
比較記憶體內容(A4L,H)和從(A1L,H)到(A2L,H),之內容,經由COUT印出不	FE36	65φ78	-458	VFY	A(Y=φ)

功　能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
同者。					
A4L,H（$42-43）加1。	& FCB4	64692	-844	NXTA4	A
A1L,H 加1，如果A2L,H小於A1L,H則設定Carry.	FCBA	64698	-838	NXTA1	A
設定GBASL,H爲列(A)。	F847	63559	-1977	GBASCALC	A
清除A暫存器成爲一 nibble（4位元）而置於低4位元內。如果進入的carry被清除，則保留A暫存器的低四位元，如果進入的carry被設定，則保留A暫存器的高4位元。	F879	636φ9	-1927	SCRN2	A
將位於（PCL,H）的指令反組合，經由COUT印出。	F8Dφ	63696	-184φ	INSTDSP	A,X,Y
計算（PCL,H）+（LENGTH），而將結果放在	F953	63827	-17φ9	PCADJ	A,X,Y

功能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
A,Y中。在叫用PCADJ之前。十進位模式的旗號必須被清除。					
讀paddle(X)到(Y暫存器)。	FB1E	64286	-1250	PREAD	A,Y
等.01秒，然後響鈴聲。	FBDD	64477	-1φ59		A,Y
放192到Y中以便得到.1秒的鈴聲。	FBE2	64482	-1φ54		A,Y
以1KHZ toggle揚聲器，持續時間爲在Y-暫存器內的週期數。	FBE4	64484	-1φ52	BELL2	A,Y
如果不爲控制字元則將字放在螢幕的復新(refresh)記憶體中。 如果爲已知的控制字元，則執行其功能。 如果爲未知的控制字元，則RTS。	FBFD	645φ9	-1φ27	VIDOUT	A,Y

功　能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
清除畫面爲空白，設定游標（cursor）在螢幕的左上方。	FC58	646φφ	-936	HOME	A,Y
將零放入Y，然後印出dash（—）。	FD9C	64924	-612		
經由COUT印出dash。	FD9E	64926	-61φ		
將字元印到螢幕輸出常規的進入點——一般爲CSWL所指之處。 對控制和控制字元採取適當的動作而將字元印到螢幕上。 如果(A)＜$Aφ跑到COUTZ，跳過反相的視頻遮蔽（inverse video mask）。	FDFφ	65φφ8	-528	COUT1	?A
當RESET鍵按下去或開電源時，監督程式的	FF59	65369	-167	RESET	

功能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
進入處。叫用SETNORM黑底白字。&					
清除65φ2 十進位模式（設定16進位）。&	FF65	65381	-155	MON	
響鈴聲。 &	FF66	65382	-154		
監督程式命令處理器的進入處。設定"＊"爲提醒字元（prompt）。&	FF69	65385	-151	MONZ	
設定(a)爲提醒字元。&	FF6B	65387	-149		
監督程式命令處理器的命令分析（parsing）常規；保留十六進位數元（digits）在A2L,H內，返回時將命令（第一個非16進位數字）放在A-暫存器，Y-暫存器設定爲下一個字元。	FFA7	65447	-89	GETNUM	

A1L,H	60,61	$3C,3D	A4L,H	66,67	$42,43	YREG	71	$47
A2L,H	62,63	$3E,3F	PCL,H	58,59	$3A,3B	XREG	70	$46
A3L,H	64,65	$40,41	ACC	69	$45			

低解析度繪圖
LORES PLOTTING

在標準的（或低解析度）畫圖模式中，螢幕的圖形區域是寬40點，高 40 點而下面 4 列爲文字或者高 48 點。X 座標是指水平位置而 Y 座標是代表垂直位置。低解析度畫圖，和文字輸出到螢幕是使用同一塊記憶區。而在圖形模式中，每一個字元位置皆代表兩個畫圖點的信息，一個緊在另一個上面。所以，在混合模式中 2ϕ 文字列被用來顯示 40 條圖形線，而在完全螢幕模式中，24 條文字列用來顯示 48 條圖表線。

每一點皆分配有 4 個位元，因此要被顯示的點可以是 16顏色的任何一種。

監督程式包含一些常規以提供下列功能：

把顯示模式設爲混合圖形和文字。

清除螢幕上的圖形部份（整個或有限部份）。

設定一個彩色控制位元組，以供每一個建立起來的畫圖點使用，其功能一直有效直到其它的彩色被選用爲止。

畫一單獨點在一個指出的垂直／水平位置。

畫一水平線從一垂直／水平點到一水平值。

畫一垂直線從一垂直／水平點到一垂直值。

將一指定座標點的彩色值傳回到要求的程式。

上面這些功能並非永遠如所預期的，有些是有限制的。舉例而言，使用設定圖形和文字混合模式的進入點，將包括清除螢幕的圖形部份，把螢幕的整個剩餘部份設定爲捲動畫面，和把游標（由現在的位置直接往下）移到螢幕的底列。另外，沒有監督程式的進入點可以設定完全螢幕的圖形模式。因此顯示模式的控制是很容易設定爲任何所希的模式，只要把適當值 poking 或貯存到適當的記憶體位置，所以這個並不是主要的問題。

使用許多第零頁的位置以完成低解析度圖形模式。

第零頁的欄位 PAGE ZERO FIELDS

欄位名	十進位位址	十六進位位址	說明
GBASL,H	38-39	$26-27	被GBASCALC設定以指向指定的畫圖線的記憶體位置。
COLOR	48	$3φ	含著所選的彩色值，此位元組的高低兩nibble（四位元的單位）均存彩色值。
MASK	46	$2E	在畫圖常規內部使用，它的值是$Fφ或$φF；依據此圖形線是在"文字"列的上半部或下半部而決定是接到的位元組之高四位元或低四位元。
H2	44	$2C	爲所畫的水平線的右端點。
V2	45	$2D	爲所畫的垂直線的底部端點。

位址表　ADDRESS TABLE

功　能	十六進位位　址	正十進位位　址	負十進位位　址	監督程式標　誌	破壞的暫存器
畫一點在列(A)，行(Y) GBASL，H 和MASK保留被設定之狀態。	F8φφ	63488	-2φ48	PLOT	A
畫一點，列由GBASL，H 和MASK 決定行在Y 中。	F8φE	635φ2	-2φ34	PLOT1	A
畫位於(A)的水平線，從(Y)到（H2），從左到右。	F819	63513	-2φ23	HLINE	A,Y
畫水平線，列由GBASL，H 和MASK指定從(Y)到（H2）。	F81C	63516	-2φ2φ	HLINE1	A,Y
畫在(Y)的垂直線，從(A)到（V2）。	F828	63528	-2φφ8	VLINE	A
畫位於(Y)的垂直線，從(A)＋1＋carry到（V2）。	F826	63526	-2φ1φ	VLINEZ	A

功能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
畫位於(Y)的垂直線，從(A)＋1到（V2）。	F82D	63533	-2φφ3		A
清除完全（48列）螢幕。	F832	63538	-1998	CLRSCR	A,Y
清除圖形區域（40線）。	F836	63542	-1994	CLRTOP	A,Y
清除部份圖形從線φ到(Y)40行寬。	F838	63544	-1992	CLRSC2	A,Y
清除部份圖形從φ線到（V2）40行寬。	F83A	63546	-199φ		A,Y
清除部份圖形左上部分，列從φ到（V2），行是φ到(Y)。	F83C	63548	-1998	CLRSC3	A,Y
從左上角到(Y)，（V2），設定LORES 螢幕爲COLOR。 進入時A-暫存器必須爲φ。 進入時Y-暫存器＝要被設定的右邊行數。 設定V2爲要被設定的最	F84φ	63552	-1984		A,Y

功　能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
後一線。					
對下面各點，設定COLOR 爲(A)。	F864	63588	-1948	SETCOL	A
改變COLOR爲(COLOR) + 3。	F85F	63583	-1953	NXTCOL	A
將位於(A)，(Y)點彩色值放入A。	F871	636φ1	-1935	SCRN	A
以A來設定GBASL,H 。(A)＝列數 / 2。	F847	63559	-1977	GBASCALC	A
設定彩色圖形顯示模式和做下面的動作；	FB4φ	6432φ	-1216	SETGR	A,Y
設定圖形模式爲混合模式。	& FB43	64323	-1213		A,Y
清除螢幕的圖形部份。	& FB46	64326	-121φ		A,Y
$14 放入A以給WNDTOP用。	& FB49	64329	-12φ7		A
將A貯放到WNDTOP 。	& FB4B	64331	-12φ5	SETWND	A
φ放入A以給WNDLFT用。	& FB4D	64333	-12φ3		A
將A貯放到WNDLFT。	& FB4F	64335	-12φ1		A
$28 放入A給WNDWDTH用。	& FB51	64337	-1199		A

功能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
將A貯放到WNDWDTH。	& FB53	64339	-1197		A
$18放入A以給WNDBTM用。	& FB55	64341	-1195		A
將A貯存在WNDBTM。	& FB57	64343	-1193		A
$17存入A以給CV用。	& FB59	64345	-1191		A
跑至TABV以設定BASL，H。					A

資料處理功能
DATA MANUPILATION FUNCTIONS

在監督程式裡有一些常規，可以由使用者的程式叫用以執行時常需用的工作。本節所描述的常規是雜碎的常規，它將資料從一個地方移到另一個地方或者轉換程式所提供信息的型式。注意有些常規是在舊型監督程式和自動開始的監督程式都有，而其它常規則僅在一種監督程式裡有。因此列出三個位址表；一個爲兩種監督程式都有，一個爲舊型監督程式才有，另一個則自動開始的監督程式才有。

常規　ROUTINES

記憶體到記憶體之間的搬移
Memory to Memory Move

這個常規是由監督程式的“M”命令使用。當命令解釋器(Command Interpreter)掃瞄鍵盤輸入，把值放入欄位A1,A2,和A4。當命令解釋器遇到“M”時，它叫用標誌爲MOVE的常規，如表中所指。從位置(A1)到(A2)的記憶體內容被移到由(A4) 開始的記憶體位置。參看“Secondary Display Area Ways and Means”的程式範例，以瞭解在BASIC中使用MOVE ，其中以監督程式的GO常規的協助來設定暫存器。

跳至某位址同時作暫存器載入

跳至某位址同時作暫存器載入
Jump to Address with Registers Loaded

在監督程式內對“G”命令反應的常規，可由BASIC 或APPLESOFT 進入，它把保留區內的值載入暫存器，然後把控制權交給PCL,H 內所指的位置，以使用其他常規。所以，一個BASIC 能夠建立起目的地的位址和暫存器的內容，然後CALL-468以進入所要求的常規。本章節內的例子程式和在“次要顯示區域Secondary Display Areas”一節中均使用這方式。

位址欄的遞增　Increment Address Fields

上面所描述的監督程式MOVE常規，是一個叫用NXTA4 和

NXTA1 常規的例子。當NXTA4 被叫用，它把兩個位元組欄位A4L,H加1，然後跳下標誌爲NXTA 1 的常規。在NXTA 1 常規把A1L,H兩個位元組欄位加1，然後在回到叫用程式之前，這欄位與A2L,H這兩個位元組欄位作比較。當返回到叫用者程式，如果（A1L,H）小於或等於（A2L,H）則 carry 狀態位元被清除。如果（A1L,H）大於（A2L,H）則 carry 被設定。

保留6502暫存器 Save 6502 Registers

SAVE 常規常被監督程式的不同常規使用，以把 65φ2 的暫存器儲存在第零頁位置 $45 - $49 。這常規在某些條件下可由使用者程式叫用　即監督程式或其它任何程式不能在同一時間叫用 SAVE 。在舊型監督程式中，SAVE 和RESTURE 被用來支援監督程式命令S 和T（即單步和追踪命令）。在兩種監督程式中，SAVE 常規在 BRK 插斷時被叫用，由 SAV1 進入，因爲在進入 IRQ 插斷處理時已把A暫存器存在 $45 了。

重新取回6502暫存器 Restore 6502 Registers

在標誌RESTURE 處的常規是 SAVE 常規的反動作，但沒有把S暫存器取回。在舊型的監督程式中，RESTORE 被指令單步和追踪常規所使用，即在每個被追踪指令執行之前使用。在兩種監督程式中，在將控制權交給操作者指的是地方前，即監督命令G 的執行中，由RESTORE 把值放回暫存器。

兩個位元組欄位的乘法　Multiply Two Byte Fields

MUL 和MULPM 常規把兩個兩位元組欄位相乘，得到一個四位元組的乘積。它們僅是比舊型的監督程式中存在。如果一個程式（如一個組合程式）叫用位於FB6φ 的MULPM，而它是在自動開始的監督程式的機械中執行，則此結果爲每次叫用時，螢幕將被清除且把"APPLE II"寫在頂列。

乘法常規　Multiply Routine

注意在下面中，乘法和除法的資料欄與其它在Apple 中的多位元組數值是相同格式：即最低的記憶體位址爲最低有效的位元組（LSB）。

設定乘數（multiplier）在	$55,54（MSB, LSB）
設定被乘數（multiplicand）在	$51 , 5φ（MSB , LSB）
應該爲零 ── 看註解	$53 , 52

依照正負號的轉換或所要求叫用 / JSR FB6φ 或FB63（-1184或-1181）（MULPM或MUL）。

依據其有效位順序，其結果是位於$53 , $52 , $51 , $50，這結果爲正。如果這兩個輸入因數中恰有一個（但不是兩個）爲負數，則SIGN（位於$2F）的$φ1位元爲 1 ，以指出在使用者程式使用此結果之前要先將其化爲補數（Complemented）。

註解：上面表中的值指出在叫用乘法之前，$53,52 應該先被設定爲零。如果沒有這樣做，則此欄的初值將被加到此結果內。

例子：

被叫用的常規	輸入				輸出				
	$51	$50	$55	$54	$53	$52	$51	$50	$2F
MULPM	ØØ	Ø1	ØØ	Ø1	ØØ	ØØ	ØØ	Ø1	ØØ
	ØØ	Ø1	Ø1	ØØ	ØØ	ØØ	Ø1	ØØ	ØØ
	Ø4	ØØ	Ø8	ØØ	ØØ	2Ø	ØØ	ØØ	ØØ
	FC	ØØ	Ø8	ØØ	ØØ	2Ø	ØØ	ØØ	Ø1
	FC	ØØ	F8	ØØ	ØØ	2Ø	ØØ	ØØ	Ø2
	7F	FF	7F	FF	3F	FF	ØØ	Ø1	ØØ
	8Ø	ØØ	Ø2	ØØ	Ø1	ØØ	ØØ	ØØ	Ø1
	8Ø	ØØ	8Ø	ØØ	4Ø	ØØ	ØØ	ØØ	Ø2
MUL	ØØ	Ø1	ØØ	Ø1	ØØ	ØØ	ØØ	Ø1	
	ØØ	Ø1	Ø1	ØØ	ØØ	ØØ	Ø1	ØØ	
	Ø4	ØØ	Ø8	ØØ	ØØ	2Ø	ØØ	ØØ	
	FC	ØØ	Ø8	ØØ	Ø7	EØ	ØØ	ØØ	
	FC	ØØ	F8	ØØ	F4	2Ø	ØØ	ØØ	
	ØØ	FC	ØØ	F8	ØØ	ØØ	F4	2Ø	
	8Ø	ØØ	Ø2	ØØ	Ø1	ØØ	ØØ	ØØ	
	8Ø	ØØ	8Ø	ØØ	4Ø	ØØ	ØØ	ØØ	
	12	34	56	78	Ø6	26	ØØ	6Ø	

將四個位元組的被除數除以兩個位元組的除數
Divide Four Byte Dividend by Two Byte Divisor

這個常規將一個四位元組的被除數除以一個兩位元組的除數，而得到一個兩位元組的商和一個兩位元組的餘數。它僅適用於舊型的監督程式。這常規是將位於 $53,52,51,5ϕ位元組的值除以位於 $55,54 位元組的值，而商放在 $51,5ϕ 內且餘數放在 $53,52中（最有效

位到最無效）。

如果 $53,52 的內容大於 $55,54 的內容，則此結果並不合乎商的條件——此結果滿溢（ overflow)。叫用程式不能讓此事發生。

由於與次方無關，將此四位元組的被乘數變成一個整數值，且將位於 $55,54 的除數亦當成一個整數值，則商和餘數欄亦爲整數。

如果DIVPM 進入點被使用，正負號可能是個問題。被乘數的正負位元爲 $51 位元組的第 $8φ 位元。如果要作的除法爲兩個位元組（在除之前，$53,52 爲 φ ），則此正負欄分別提供，而正負號位元爲 $2F 的LSB。如果是叫用 DIVPM，且如果 $2F 含著 $φ1 ，則在使用其結果之前，必須將它化爲補數。

當使用沒有正負號的除法，其進入點爲 DIV ，則此除法爲32位元欄除以16位元欄，而得到16位元的結果。

例子：

被叫用的常規	輸入		輸出		
	被除數	除數	商	餘數	正負號
	$53 52 51 50	$55 54	$51 50	$53 52	$2F
DIVPM	ØØ 4Ø ØØ ØØ	Ø8 ØØ	Ø8 ØØ	ØØ ØØ	ØØ
[$FB81]	ØØ ØØ ØØ Ø8	ØØ Ø4	ØØ Ø2	ØØ ØØ	ØØ
[64385]	ØØ Ø1 ØØ ØØ	ØØ Ø2	8Ø ØØ	ØØ ØØ	ØØ
[-1151]	ØØ ØØ ØØ Ø3	ØØ Ø2	ØØ Ø1	ØØ Ø1	ØØ
	ØØ ØØ 3Ø ØØ	Ø2 ØØ	ØØ 18	ØØ ØØ	ØØ
	ØØ ØØ 3Ø ØØ	2Ø ØØ	ØØ Ø1	1Ø ØØ	ØØ

	ØØ ØØ 33 33	ØØ 22	Ø1 81	ØØ 11	ØØ
	ØØ 1Ø 4Ø ØØ	Ø4 ØØ	Ø4 1Ø	ØØ ØØ	ØØ
	ØØ 2Ø 8Ø ØØ	Ø8 ØØ	Ø4 1Ø	ØØ ØØ	Ø1
	ØØ 2Ø 82 ØØ	Ø8 ØØ	Ø4 ØF	Ø6 ØØ	ØI
	ØØ 1Ø 41 ØØ	Ø4 ØØ	Ø4 1Ø	Ø1 ØØ	ØØ
DIV					
[$FB84]	ØØ 8Ø ØØ ØØ	8Ø ØØ	Ø1 ØØ	ØØ ØØ	
[64388]	ØØ ØØ 8Ø ØØ	Ø8 ØØ	ØØ 1Ø	ØØ ØØ	
[-1148]					

建立RESET向量 Establish a RESET Vector

自動開始監督程式提供一個位址向量以便完成一個RESET 插斷的處理。它被稱之爲軟性入口向量（ Soft Entry vector ），因爲它的設計可允許在RESET 後繼續原有的處理。這個向量是位於第三頁中，且含著一個位址；當螢幕，鍵盤和其它基本的Apple硬體項目被設定爲它們的"起始"狀態時，控制權便轉移至此位址。舉例而言，顯示硬體被設定爲顯示主要區域的文字，且設定爲全螢幕的捲動畫面。

在如此一個起動被執行後，位置$φ3F3 和$φ3F4 再度地彼此測試，以決定位於$φ3F2-$φ3F3 的向量是否被認爲有效。如果是，則控制權將被轉移至（$φ3F2-φ3F3 ）。一般而言，這個結果是將控制權傳遞給$Eφφ3 ，以完成進入到監督程式內控制C的動作，即再進入BASIC或APPLESOFT。在靴帶動作時，DOS將它自己的重新開始（ restart ）點放在這向量中。當然啦，你可能希望在這向量中設定其他值，如將造成監督程式（具有提醒星號 asterisk）被叫用，在舊型監督程式的正常情況便是如此。爲了要在那向量中設定一個不同

的值，POKE 或貯存所希望的值至 $φ3F2-$φ3F3 中，然後CALL或JSR 到 SETPWRC（ $FB6F 或 -1169 ）以使得監督程式適當地設定 $φ3F4 。

把十六進位字元轉換爲數值以供使用
Convert Hex Characters to Value for Use

程式設計者的公用程式常需要輸入 16進位的位址或資料，而不是十進位。此監督程式亦使用16進位的輸入，因此有一種方法把輸入的16進位字元轉換爲數值存在某欄內。在監督程式中的GETNUM 常規把由鍵盤輸入區域（ $φ2φφ-$φ2FF ）來的字元轉換16進位值，此數值貯存在A2L,H和在有些條件下是貯存在A1L,H和A3L,H。

GETNUM常規轉換位於 $φ2φφ 區域內的字元，由位置 $φ2φφ+（Y-暫存器）開始，且一直繼續執行到發現了不爲十六進位數元（非φ-9 或A-F）爲止。如果這字串超過 4 個16進位數元，則置於A2L,H（如果（MODE）＝φ，則亦置於A1L,H和A3L,H中）中的結果則爲轉換字串中的最後 4 個16進位數元。如果這字串是少於 4 個16進位數元則存結果的欄內含著的值爲向右對齊而前面爲零。在本節結尾處的例子程式顯示如何從APPLESOFT 使用GETNUM。

指令的反組合　Disassemble an Instruction

Apple II 監督程式含有一個反組合程式，憑這程式使用者可以用

助憶符號（mnemonics）來顯示一部份的機械語言程式，而不是僅顯示16進位數字而已。當監督程式命令"L"被使用時，控制權交到標誌爲LIST（$FE5E）的常規，此常規設定一計數器爲2 ϕ，然後叫用"單一指令反組合程式"20次，其中適當地調整指令指標（pointer）PCL,H。這個常規可以被當做一個例子，以了解如何使用位於位址表中標誌爲INSTDSP和PCADJ的常規。

INSTDSP常規利用INSDS1常規，對位於（PCL,H）的指令適當地設定第零頁內之位置FORMAT和LENGTH。INSDS1亦將PCL,H的內容印到螢幕上，即是要被反組合指令的位址。當從INSDS1返回時，INSTDSP常規控制此反組合列的其餘部份之印出。

注意PCL,H並不會受反指令的反組合而改變。所以，它必須被叫用INSTDSP的程式所維護。這是經由叫用PCADJ常規來完成，此常規利用PCL,PCH和LENGTH（由INSDS1設定）計算出新值置於A-暫存器和Y-暫存器，將分別放入PCL和PCH，如此將新值傳回給叫用它的程式。

位址表 ADDRESS TABLE

功能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
僅適用於舊型的監督程式					
乘以有正負號的欄位，把正負	FB6ϕ	64352	-1184	MULPM	A,X,Y

功　能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	被破壞的暫存器
號留在SIGN的LSB。					
乘以沒有正負號的欄位，(51,5φ) * (55,54)=(53,52,51,5φ)。	FB63	64355	-1181	MUL	A,X,Y
除以有正負號的欄位，把正負號留在SIGN的LSB(從51,55)。	FB81	64385	-1151	DIVPM	A,X,Y
沒有正負號的欄位作除法(53,52,51,5φ)/(55,54)=(51,5φ)。	FB84	64388	-1148	DIV	A,X,Y
設定ACL,H和AUXL,H的絕對值，使得結果的正負號放在SIGN的LSB(被MULPM和DIVPM D4叫用)。	FBA4	6442φ	-1116	MD1	A,X,Y

功能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
僅適用於自動開始的監督程式					
設定RESET向量的有效值	FB6F	64367	-1169	SETPWRC	A
同時適用於舊型和自動開始的監督程式					
監督程式命令處理器GO的進入處。 如果進入的話，用A1L,H設定PCL,H。	FEB6	652φ6	-33φ	GO	A,X,Y,P
叫用RESTORE，除了S外設定所有暫存器。	FEB9	652φ9	-327		
透過PCL,H作跳越。	FEBC	65212	-324		
移動記憶體內的位元組，把（A1L,H）至（A2L,H）移到（A4L,H）。 注意：在進入時Y-暫存器必須爲零。	FE2C	65φ68	-468	MOVE	A
指標(pointer)&A4L,H加1。	FCB4	64692	-844	NXTA4	A
指標(pointer)	FCBA	64698	-838	NXTA1	A

功　能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
A1L,H 加1，如果(A1L,H)是大於(A2L,H)則設定 carry。					
把65φ2 暫存器A,X,Y,P,S保留在$45-$49。	FF4A	65354	-182	SAVE	A,X
從$45-$48 處重新取回65φ2的暫存器A,X,Y,P。	FF3F	65343	-193	RESTORE	A,X,Y.P
把$200，Y位置的十六進位字元轉換爲數值放入A2L,H（若(MODE)＝φ，並放入A1L,H和A3L,H)'。	FFA7	65447	-89	GETNUM	A,X,Y
反組合一指令並經由COUT顯示出來。	F8Dφ	63696	-184φ	INSTDSP	A,X,Y

功能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
在反組合或追踪或單步之後一將結果放在A，Y暫存器中以便作爲(PCL，H)。	F953	63827	-17φ9	PCADJ	A,X,Y

Applest資料處理程式範例

APPLESOFT SAMPLE DATA MANIPULATION PROGRAM

```
10          REM  DATA MANIPULATION FUNCTION
20          REM   SAMPLE PROGRAM
30          REM  MEMORY DUMP
40          REM   OF HEX AREA INDICATED.
50    GOTO 1000:          REM  BYPASS SUBROUTINES
200         REM   CALL GETNUM ROUTINE VIA GO ROUTINE
210   POKE 58,167:        REM  PCL=$A7
220   POKE 59,255:        REM  PCH=$FF
230   SI$ = AD$ + " ":    REM  BUILD STRING TO STORE
240   FOR I = 1 TO  LEN (SI$)     REM:  STORE STRING IN INPUT BUFFER
250   CC$ =  MID$ (SI$,I,1)       REM:                ''
260   CC% =  ASC (CC$) + 128      REM:                ''
270   POKE 512 + I,CC%
280   NEXT
290   POKE 71,1:          REM  SET YREG TO START AT LOCATION 513
300   POKE 49,0:          REM  CLEAR MODE BYTE
310   CALL  - 327:        REM  GO PROCESSOR
320   ST =  PEEK (62) + 256 *  PEEK (63):   REM  ST=START ADDRESS($A2)
330   IF ST > 32767 THEN ST = ST - 65536    REM  TWO'S COMPLEMENT
                                                 ADDRESS IF >= $8000
340   RETURN
```

```
600           REM  DISPLAY HEX CONTENTS
610   SH% = ST / 256      REM GET HI ADDRESS BYTE
620   SL% = ST - SH% * 256:  REM  GET LO ADDRESS BYTE
630   IF SH% < 0 THEN SH% + 256:  REM  GET 2'S COMP IF NECESSARY
640   POKE 60,SL%:POKE 61,SH%
650   RM% = SL% - ( INT (SL% / 8 )) * 8  REM  RM% = MOD 8 OF LO BYTE
660   IF RM% THEN CALL -622
670   POKE 71,0:  REM  SET "Y" REG TO ZERO
680   POKE 58,163:    REM  PCL = $A3
690   POKE 59,253:    REM  PCH = $FD
700   CALL  - 327:    REM  CLEAR "Y" REG & $FDA3G
710   POKE 36,29:  PRINT "! ";:  REM  SEPARATES HEX FROM ASCII
720           REM  DISPLAY ASCII CHARACTER CONTENTS
730   SE = ST + 7 - RM%  REM  SEPARATES HEX FROM ASCII
740   FOR I = ST TO SE   REM  PRINT ASCII CONTENTS
750   CX =  PEEK (I): IF CX < 128 THEN CX = CX + 128
760   CX$ =  CHR$ (CX): IF CX < 160 THEN CX$ = "?"
770   PRINT CX$;
780   NEXT
790   RETURN
1000          REM  PROGRAM START
1010  PRINT "HEX DISPLAY"
1020  INPUT "ENTER ADDRESS ";AD$
1030  IF AD$ = "END" THEN  END
1040  IF  LEN (AD$) = 0 THEN 1100:REM CONTINUE WITH NEXT AVAILABLE
                                        ADDRESS
1050  GOSUB 200:
1080  FOR J = 1 TO 16:    REM  PRINT 16 LINES
1090  GOSUB 600
1100  ST = ST + 8 - RM%
1110  NEXT
1120  PRINT
1130  GOTO 1020
```

監督程式的命令處理器
MONITOR COMMAND PROCESSOR

監督程式命令處理器爲監督程式的一部份，此對在“ * ”提醒字元時打入的命令有所反應。這些命令包括將資料從一個位置移到另一

個位置，磁帶的讀和寫，指令的反組合，和其它在參考手册中所描述的。參考手册含著一份使用命令的完整說明。本書在這節中描述如何從一個使用者的程式來叫用某些命令處理常規，而不是從鍵盤輸入而叫用，且跳入監督程式內而不返回到使用者程式。

進入監督程式的命令處理器
ENTERING THE MONITOR COMMAND PROCESSOR

監督程式的命令處理器爲監督程式的一部份，它在星型的提醒（asterisk prompt）字元時來讀鍵盤的輸入且執行所要求的服務。"進入"命令處理器隱含著將機器的控制權移至監督程式模式。在舊型監督程式中，當按下RESET 鍵，則電腦是在監督程式模式。在自動開始監督程式的機器中，當按下RESET 鍵，電腦一般都跑至BASIC或APPLESOFT 中。在自動開始監督程式中，要跳至監督程式的僅有方法爲叫用這些進入點的其中之一（一般都是CALL-151 ）。

在這個模式中，在記憶體裡的資料可以利用監督程式MOVE命令來移動。可以經由卡式磁帶的資料傳遞命令，從磁帶讀取整堆的資料。或者任何其它的監督程式命令都可以使用。也就是已進入監督程式模式後，監督程式命令處理器就從鍵盤讀取命令，然後再根據它們來動作。

在位址表中指出一些進入點，以便從這些進入點"進入"監督程式命令處理器。請注意，一旦此監督程式被跳至特定的點，所有在此

點之後所描述的起動動作都將被執行。這在每個功能描述結尾處以“&”來註明。

叫用監督程式的命令處理器
CALLING THE MONITOR COMMAND PROCESSOR

“叫用”監督程式的命令處理器隱含著“Return”將回到叫用程式。無論如何，監督程式命令處理器的驅動部份並沒有設計成以這種型式來作業，所以需要一個短的機器語言程式，以便在離開後是返回到叫用程式。在本節的結尾處，提供一例子程式，指出所需的動作。在此例子中，有三個位元組的機器語言程式放在第二頁中（位於$φ2FC），但是它可以擺放在所希望的任何位置。有了這個程式，從BASIC或APPLESOFT叫用監督程式可以應用。

一個叫用命令處理器的程式首先必須在某個地方貯存三個位元組的離開程式。然後此程式可以POKE一串的監督程式命令到輸入區域，由位址$φ2φφ 開始，每一串中的最後一個命令爲一監督程式的GO 命令，以把控制權交給離開程式。在例子中，在命令串中最後的監督程式命令爲“φ2FCG”。離開程式（exit routine）的功能爲從堆疊器（stack）中取出（pull）一個返回位址（兩個位元組），然後做一個RTS以返回到BASIC，APPLESOFT 或機器語言叫用程式。

位址表 ADDRESS TABLE

功能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
監督程式命令處理器，"CR"所使用的"空白"進入點。	FEφφ	65φ24	-512	BL1	A,X,Y
監督程式命令處理器，"空白"命令進入點。	FEφ4	65φ28	-5φ8	BLANk	A,X,Y
監督程式命令處理器，貯存常規。	FEφB	65φ35	-5φ1	STOR	A
監督程式命令處理器，對逗點，句點，正號或負號設定MODE。	FE18	65φ48	-488	SETMODE	A,Y
貯存適當的值至MODE，亦從BLANK進入。	FE1D	65φ53	-483	SETMDE	none
監督程式命令處理器對小於（<）的命令。	FE2φ	65φ56	-48φ	LT	A,X
監督程式命令處理器MOVE常規。位置（A1L,H)至(A2L,H)被移到由(A4L,H	FE2C	65φ68	-468	MOVE	A(Y=φ)

功　能	十六進位位　址	正十進位位　址	負十進位位　址	監督程式標　誌	破壞的暫存器
）開始位置。					
監督程式命令處理器VERIFY常規。(A1L,H）到(A2L,H）與由（A4L,H）位置開始作CL 後，不同的則經由COUT 印出。	FE36	65φ78	-458	VFY	A(Y=φ)
監督程式命令處理器LIST（反組合程式）常規：經由COUT 顯示20個指令。	FE5E	65118	-418	LIST	A,X,Y
設定 INVFLG爲$3F=反相的視頻。	FE8φ	65152	-384	SETINV	Y
設定 INVFLG爲$FF=一般的視頻。	FE84	65156	-38φ	SETNORM	Y
以Y暫存器來設定 INVFLG。	FE86	65158	-378	SETIFLG	none
設定埠φ（鍵盤）爲輸入。	FE89	65161	-375	SETKBD	A,X,Y
設定埠(A)爲輸入	FE8B	65163	-373	INPORT	A,X,Y
設定埠（A2L）爲輸入。	FE8D	65165	-371	INPRT	A,X,Y

功能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
設定埠φ（螢幕）爲輸出。	FE93	65171	-365	SETVID	A,X,Y
設定埠(A)爲輸出	FE95	65173	-363	OUTPORT	A,X,Y
設定埠（A2L）爲輸出。	FE97	65175	-361	OUTPRT	A,X,Y
監督程式命令處理器GO的進入點。 如果進入則以&A1L,H 設定PCL,H。	FEB6	652φ6	-33φ	GO	A,X,Y,P
叫用RESTO-RE,設定除S外的所有暫存器。	FEB9	652φ9	-327	&	
透過PCL,H作跳越。	FEBC	65212	-324		
監督程式命令處理器顯示暫存器內容。	FEBF	65215	-321	REGZ	
監督程式命令處理器 carriage Return 進入處。 首先，模擬空白的輸入。 然後，從堆疊器中 POP 兩次且跑至位於MONZ處的監	FEF6	6527φ	-266	CRMON	

功　能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
督程式命令處理器。					
從$45-49 處重新取回暫存器：把STATUS載入A暫存器且壓入堆疊器上。	FF3F	65343	-193	RESTORE	
把 ACC 放入到A。	FF42	65346	-19φ		
把XREG放入到X。	FF44	65348	-188	RESTR1	
把YREG放入到Y。	FF46	65350	-186		
由堆疊器內取回P暫存器（PLP）且RTS。	FF48	65352	-184		
保留65φ2 暫存器於$45-49。保留A暫存器在ACC$45。	FF4A	65354	-182	SAVE	
保留X暫存器在XREG$46。	FF4C	65356	-18φ	SAV1	
保留Y暫存器在YREG$47。	FF4E	65358	-178		
保留P暫存器在STATUS$48。	FF5φ	6536φ	-176		

功能	十六進位位址	正十進位位址	負十進位位址	監督程式破壞的暫存器標誌
保留S暫存器&在SPNT$49。	FF54	65364	-172	
清除65φ2 十進位模式(設定16進位)。				
當RESET 鍵壓下或開電源時監督程式的進入點。	FF59	65369	-167	RESET
叫用SETNORM&一黑底白字。				
叫用 INIT 一文字+完全螢幕。	FF5C	65372	-164	
叫用 SETVID&一螢幕爲輸出。	FF5F	65375	-161	
叫用SETKBD &一鍵盤=輸入。	FF62	65378	-158	
清除65φ2 十進&位模式(設定16進位)。	FF65	65381	-155	MON
響鈴聲。 &	FF66	65382	-154	
監督程式命令處理器進入點,設定"*"爲&提醒字元。	FF69	65385	-151	MONZ

功　能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
設定(A)爲提醒&字元。	FF6B	65387	-149		
叫用GETLNZ以&讀取命令列。	FF6D	65389	-147		
在掃瞄列之前清&除MODE。	FF7φ	65392	-144		
拿取(pick up)一個命令：叫用GETNUM以掃瞄輸入列，保留16進位數元在A2L,H，且以非16進位形式放在A暫存器中返回。把Y保留在YSAV——現目前命令列的位置。	FF73	65395	-141	NXTITM	
叫用非十六進位字所指的常規，此非十六進位字由GET-NUM傳回。	FF82	6541φ	-126		
從監督程式命令服務常規返回；從YSAV處取回Y。這是	FF85	65413	-123		

功能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
如果還有其它命令的話，則跑至NXTITM其處理列上的下方命令。					
監督程式命令處理器的分析常規；保留16進位數元在A2L，H中，返回時命令（第一個非16進位字）放在A暫存器內，設定Y暫存器以便得到下一個字元。	FFA7	65447	-89	GETNUM	
叫用命令字所指的常規： 把位址$FEXX壓入堆疊器上。 （MODE）放入A暫存器內傳給被叫用程式。 在叫用之前清除MODE。 用RTS叫用所選的常規。	FFBE	6547ϕ	-66	TOSUB	

功　能	十六進位位址	正十進位位址	負十進位位址	監督程式破壞的標誌暫存器
在命令之間清除MODE位元組。	FFC7	65479	-57	ZMODE
僅適用於舊型監督程式（OLD MONITOR ONLY）				
執行位於（PCL,H）的指令，且顯示此指令和暫存器的結果。	FA43	64φ67	-1469	STEP
監督程式命令處理器TRACE指令常規。	FEC2	65218	-318	TRACE
監督程式STEP一個指令。	FEC4	6522φ	-316	STEPZ

A1L,H	6φ,61	$3C,3D	PCL,H	58,59	$3A,3B
A2L,H	62,63	$3E,3F	ACC	69	$45
A3L,H	64,65	$4φ,41	XREG	7φ	$46
A4L,H	66,67	$42,43	YREG	71	$47
YSAV	52	$34			

Applesoft程式範例

APPLESOFT SAMPLE PROGRAM

```
1           REM  MONITOR COMMAND PROCESSOR SAMPLE PROGRAM
10 AA$ = "2FC:68 68 60 N 2FCG ": REM SET UP RETURN ROUTINE @2FC
11    GOSUB 1000:       REM  MOVE COMMAND TO KEYBOARD INPUT AREA

100                   REM  RETURN IS SET. NOW CALL
101                   REM  SOME MONITOR COMMANDS.
110   AA$ = "F800L 100.1FF 2FCG "
120   CALL  - 936:    REM  CLEAR THE SCREEN
130   GOSUB 1000:     REM DO DISASSEMBLY, MEMORY DISPLAY, RETURN
140   PRINT : PRINT :
141   PRINT "THATS ALL. "
150   END

1000  B = 511:  REM FOR LOOP IS 1 TO LIM, SO B=BYTE BEFORE $200
1005  LIM =  LEN (AA$)
1010  FOR I = 1 TO LIM
1020  P$ =  MID$ (AA$,I,1)
1050  P =  ASC (P$) + 128
1070  POKE B + I,P
1080  NEXT
1085  CALL  - 144
1090  RETURN
```

經由監督程式使用喇叭
SPEAKER USE THROUGH THE MONITOR

有很多種方式可以使用Apple II中的喇叭。其中一個是代表程式事件的訊號。監督程式內含有一個常規負責這件事，它是以1 khz ，持續0.1 秒時間來觸發喇叭。這是一個聽得見的"嗶 beep"聲，當按下RESET 鍵或者完成一個磁帶記錄的讀或寫時，皆可聽到。

並不是僅Apple II 有喇叭，某些接到Apple II 的印表機（ printer ），當接到BELL碼時，也會產生某種型式的響聲。在Apple Ⅱ的鍵盤控制G鍵是BELL。這字元碼爲$87或者十進位的135。經由COUT印出這個字元將造成Apple產生嗶聲，如果印字機有此裝置亦會造成印表機的"鈴聲"發出。

有兩者方法讓使用者程式叫用監督程式裡面的這常規　　此常規對於輸出$87的反應爲發出嗶聲。

如果你想要讓Apple中響鈴聲，而不管使用的輸出裝置爲何，則直接叫用在監督程式中會產生響聲的常規；CALL-1φ59（或者CALL 64477 ）或者JSR FBDD 而A暫存器和Y暫存器內容將會被破壞。

如果螢幕爲一印字裝置，而你想要響Apple II的鈴聲，或者響印字機內的喇叭，則叫用在監督程式的進入點，它將$87放在A暫存器內且經由COUT "印出"；CALL-198（或CALL 65338 ）或JSR FF3A而A暫存器的內容將被破壞。

位址表　ADDRESS TABLE

功　能	十六進位位　址	正十進位位　址	負十進位位　址	監督程式標　誌	破壞的暫存器
如果(A)= $87，等.01秒，然	FBD9	64473	-1φ63	BELL1	A,Y

功能	十六進位位址	正十進位位址	負十進位位址	監督程式標誌	破壞的暫存器
後響"鈴聲"。否則的話，RTS。					
等.01秒，然後響鈴聲。	FBDD	64477	-1φ59		A,Y
定Y＝192以便&產生.1秒的響聲。	FBE2	64482	-1φ54		A,Y
以1KHZ觸發喇叭，此動作持續時間爲Y暫存器內的週期數。	FBE4	64484	-1φ52	BELL2	A,Y
經由COUT印出"ERR"和鈴聲碼。	FF2D	65325	-211	PRERR	A
經由COUT印出鈴聲碼（$87）。	FF3A	65338	-198	BELL	A

卡式磁帶的輸入和輸出
CASSETTE TAPE INPUT AND OUTPUT

在監督程式中有關讀和寫磁帶，有兩個主要的進入點。它們是READ和WRITE。叫用這些所需的動作將在下文中描述。有一些

其它常規進入點，由監督程式使用，其工作是以位元和位元組爲基礎。這些將在監督程式中的延長位置處描寫且指出那個Apple II程式叫用它們，但是在本手册中並不注重在連續叫用之間精確的指令時序。

截至目前，你將發現某些磁帶檔案是由一個記錄（ record）組成，而有些則由兩個 record 組成。舉例而言，LOADing 一個APPLESOFT 或BASIC程式將產生兩個鈴聲，以表示完成了從磁帶讀兩個分開的 records 。

定義：

一個磁帶 record 爲一串連續的位元，當作一個單元從記憶體被讀入或寫出。磁帶 record 爲一實際的個體。

在磁帶上的一個檔案爲一串或一系列一個或一個以上的 record，它們所含的資料有某種邏輯結構。一個檔案爲一邏輯的個體。

一個APPLESOFT 或BASIC 程式檔案含有兩個 records 。以BASIC 而言，第一個 records 爲兩個位元組長，且含著第 2 個record 的長度。當監督程式已經滿足了第一 record 的BASIC 讀取時，BASIC 使用此 record 所指的 record 長度，以決定出在記憶體內用來讓監督程式讀入第 2 個 record 的開始和結束點。每個叫用到監督程式中的READ或WRITE 僅完成一個 record的輸入或輸出工作而已。

APPLESOFT 程式被 SAVE 兩個 record組或檔案。而其第一個 record 爲二個位元組長：前兩個位元組指出長度，而第三個位元組被設定爲 $55 以表示此爲一普通的APPLESOFT II（與APPLESOFT I 不同）程式。

某些其它程式的第一個 record 較長（但固定長度），其包含檔案第 2 個 record 的長度，和其它關於檔案的信息如產生的時間或檔案的名稱。

寫 WRITE

$FECD 65229 -3φ7

在進入這點之前，在A1L,H（$3C-3D）內設定第一個位元組位址且在A2L,H（$3E-3F）設定最後一個位元組位址。監督程式將先寫10秒鐘的連續聲調（前導 header ），後再跟著所指的記憶體內容，再跟著爲一個位元組的 checksum （將所有寫到磁帶的資料做一個互斥或（XOR）的結果）。

讀 READ

$FEFD 65277 -259

在進入這點之前，在A1L,H（$3C-3D 設定第一個位元組的位址且在A2L,H（$3E-3F ）設定最後一個位元組位址。監督程式從磁帶讀入資料，將它貯存在所指定的記憶體中，且隨時維持一個執行中互斥或的結果，存於第零頁內名爲 CHKSUM（$2E）的欄位中。當最後一個所指的記憶體位置，已經被從磁帶來的資料填入時，監督程

式再讀一個位元組且與CHKSUM的內容相比較。如果相等，則監督程式響一個嗶聲，然後回到叫用程式。如果不相等，則監督程式再響一個嗶聲和返回之前，經由COUT印出" ERR "。

如果你要叫用程式決定出讀磁帶的動作是否成功，則必須做某些特殊動作。有一方法爲比較讀磁帶前後CH（$24）的內容。如果它們是相等，則並沒有印 ERR 到螢幕上。如果游標（cursor）的水平位置（CH）經過叫用READ以後改變了，則ERR 必定已被寫到螢幕上。如果遇到這種情況，則程式要求操作者者，調整磁帶位置和通知程式再嚐試讀取 record 警告：如果CSWL,H指向一個印表機卡(printer card)或者其它並不輸出到螢幕的常規，則輸出" ERR "並不含增加CH。

卡式磁帶輸入／輸出的內部常規
CASSETTE INPUT/OUTPUT INTERNAL ROUTINES

下面的進入點／常規之功能將被描述，但是對使用者的叫用其註明並不夠詳細。因爲其中有些對時序要求是相當重要的(critical)，且如何使用它們的註解，是依它們如何被使用而決定的。

HEADR

$FCC9　64713　-823

這個常規寫出同步的單音（monotone），它是每個磁帶 record 的開頭部份。當WRITE 常規叫用HEARD ，它把 $4φ 放入A暫存器

內，以造成寫入10秒鐘的前導波。此READ常規亦叫用HEARR 以便從第一次偵測出資料是從磁帶進來開始延遲，到第一次讀取作 ϕ / 1 的偵測開始爲止。READ在叫用HEADR之前，放$16到A暫存器內，所以對硬體設定的延遲被設定約 3.5 秒。這個常規並不能由BASIC或APPLESOFT叫用，但是它可由程式設計員的Aid #1磁帶驗證常規（ Programmer's Aid #1 Tape Werify routine ）使用，此常規讀取磁帶資料且與記憶體比較而不是貯存資料到記憶體。

RD2BIT

$FCFA 64762 -774

這個常規造成遞減Y暫存器的廻路，直到硬體已經指出磁帶輸入暫存器有兩次變動（ transitions ）。RDBIT常規被叫用兩次便是爲這個目的。Y暫存器在返回時的內容與進入時的內容比較，以指出兩次變動的所費的時間。

這常規在監督程式內由READ常規叫用，以延遲進入資料傳遞模式直到磁帶輸入是有用的爲止。READ叫用HEADR以便延遲 3.5 秒回返時再叫用RD2BIT。這個常規亦可被APPLESOFT ，和在程式設計員的Aid #1中的磁帶驗證（ tape verify ）和形狀表載入（ shape Table Load ）程式叫用。

RDBIT

$FCFD 64765 -771

這常規當在測試磁帶輸入暫存器，是否有從零到壹或者壹到零的

變動時，在作遞減Y暫存器的廻路。位元值零或壹可以從Y暫存器中剩餘的計數值（residual count）決定出來。這常規被監督程式內的RD2BIT和READ常規叫用。它亦可被Programmer's Aid #1 Tape Verify叫用。

RDBYTE

$FCEC　64748　-788

這個常規當需要時叫用RD2BIT 以便組合從磁帶來的一位元組的信息。然後它返回到叫用者且將位元組放在A暫存器內。除了被監督程式的READ常規叫用外，它亦被Shape Table Load（形狀表載入）in Programmer's Aid #1叫用。

WRBIT

$FCD6　64726　-810

這個常規當被HEADR常規或者WRBYTE常規叫用時，完成寫一個位元到磁帶上的動作。

WRBYTE

$FEED　65261　-275

當這常規被叫用以寫一個位元組到磁帶時，它利用WRBIT 寫十個位元到磁帶。它唯一的叫用者是監督程式內的WRITE。

PADDLES, BUTTONS & ANNUNCIATOR I/O

Apple II 有一個Game I/O連接器，它由硬體支援因而有 4 個數位輸出（ digital outputs ），三個數位輸入和四個類比輸入（稱之爲 paddles ）。監督程式要讀此 paddle 是寫一個觸發信號，以起動 paddle 的計時器（ timer ），然後讀取所選的 paddle 計時器，遞增Y暫存器直到計時器變成爲眞；此讀到的結果是在Y暫存器內。監督程式提供數位輸出或數位輸入是不需要的。要存取（ access ）數位的I/O埠可以經由 PEEKing 或 POKEing 適當的位址；或如果使用機械語言，則爲 LDx 或 STx。自動開始的監督程式當任何 RESET 鍵插斷時，設定數位輸出埠（ annunciators ）的最初狀態。經由參用位址 C\phi$58 和 C\phi$5A， ANϕ 和 AN1 被設定爲清除（TTL LO）狀態。經由參用位址 C\phi$5D 和 C\phi$5F ，AN2 和 AN3 被設定爲設定（set）（TTL HI）狀態。

使用監督程式所提供的東西以讀取 paddle 的設定狀態，可用 JSR 到

PREAD	FB1E	64286	-125ϕ

以 paddle 號碼（ϕ - 3 ）放於X暫存器內，當返回時 paddle的"值"將被發現在Y暫存器內。在這過程中，A暫存器內容被破壞。（APPLESOFT和BASIC提供讀取 paddle 的動作，所以在那裡並不

需要設定X和查看Y。）

直接讀取 paddles 可以經由讀取 paddle 觸發器（ trigger ），以起動所有 paddle 計時器，然後當計數時，不斷地讀取適當的 paddle 輸入位址，直到從 paddle 位址讀進來的值不再是 $8φ 位元爲被設定狀態爲止。

警告：當讀了一個 paddle 後，在讀其它的 paddle之前要空一段時間否則可能有不正確的結果。當此 paddle 觸發器被觸發時，所有的計時器開動。如果你讀的第一個 paddle 爲-- low 值，在迅速返回以讀取另一個 paddle 時，你所看到的變動（ transition ）是從第一個 paddle 觸發器而來而不是第二個。參看（控制Y與參數的使用）" Use of Control-Y with Parameters "一節中的例子程式。另一個解法爲在眞正的讀之間，做一個讀假的 paddle（ fake paddle ）的動作。

GAME I/O的硬體位址表

Game I/O 硬體位址	十六進位位址	正十進位位址	負十進位位址	動作 / 註解
起動 paddle 計時器。	Cφ7φ	49264	-16272	
paddle φ 計時器。	Cφ64	49252	-16284	負值到計時器終
paddle 1 計時器。	Cφ65	49253	-16283	止。
paddle 2 計時器。	Cφ66	49254	-16282	
paddle 3 計時器。	Cφ67	49255	-16281	

Game I/O 硬體位址	十六進位位址	正十進位位址	負十進位位址	動作/註解
paddle φ 開關。	Cφ61	49249	-16287	
paddle 1 開關。	Cφ62	4925φ	-16286	負值表示按鈕。
paddle 2 開關。	Cφ63	49251	-16285	(button)壓下
(Annunicator 爲數位輸出埠)				
清除 Annunciator φ 輸出。	Cφ58	4924φ	-16296	POKE/Store
設定 Annunciator φ 輸出。	Cφ59	49241	-16295	零到適
清除 Annunciator 1 輸出。	Cφ5A	49242	-16294	當的位
設定 Annunciator 1 輸出。	Cφ5B	49243	-16293	址
清除 Annunciator 2 輸出。	Cφ5C	49244	-16292	
設定 Annunciator 2 輸出。	Cφ5D	49245	-16291	
清除 Annunciator 3 輸出。	Cφ5E	49246	-1629φ	
設定 Annunciator 3 輸出。	Cφ5F	49247	-16289	

等待常規
WAIT ROUTINE

此WAIT 常規由一個廻圈內含另一個廻圈所構成，由於這種結構，所以在廻路中所花費的時間是與進入時的A 暫存器值成幾何的變化。叫用這個常規將會造成一個可預定時間長度的廻路，故如在監督程式關於使用喇叭爲一鈴聲時即可叫用此常規。另外，它在寫資料到一個低速的裝置如印字機或一個打字機時也是相當有用的。

WAIT	$FCA8	5468φ	-856

分析其程式碼可看出在叫用WAIT (JSR) 和到WAIT 結尾處的RTS 之間的時間約爲

2.5A **2 + 13.5 *A + 13 個 1.φ23 微秒的機器週期。其中 A 爲累積器的內容。

另外一種表示爲

TIME IN MICROSECONDS = (2.5 *(A 2) + 13.5 *A + MC) *MS

其中　A = 累積器的內容

MC = 13 個機器週期

MS = 1.φ23 微秒

下面這表指出進入時的 A 暫存器值在WAIT 常規所產生的延遲時間。

WAIT常規的延遲時間

A-reg (Dec.)	Time in seconds	A-reg (Dec.)	Time in seconds	A-reg (Dec.)	Time in seconds
1	.000029667	49	.006830571	137	.049907055
2	.00005115	50	.007097574	138	.050624178
3	.000077748				
4	.000109461	53	.007929273	150	.059628624
5	.000146289	54	.008216736	151	.060412242
6	.000188232	55	.008509314		
7	.00023529	56	.008807007	162	.06936963
8	.000287463	57	.009109815	163	.070214628
9	.000344751	58	.009417738		
		59	.009730776	174	.079847196
17	.000987195	60	.010048929	175	.080753574
18	.001090518				
19	.001198956	73	.014650383	184	.089141151
		74	.015040146	185	.090098679

25	.001956999	75	.015435024		
26	.002101242			195	.099955284
		85	.019665129	196	.100969077
31	.002899182	86	.020116272		
32	.003074115			204	.109263561
		96	.024909027	205	.110323389
36	.003824997	97	.025416435		
37	.004025505			218	.124566618
		105	.029659839	219	.125698056
41	.004878687	106	.030213282		
42	.00510477			239	.149400966
		122	.03976401	240	.150639819
45	.005813709	123	.040404408		
46	.006060252			255	.169836414

控制—Y與參數的使用
USE OF CONTROL-Y WITH PARAMETERS

在APPLESOFT的手册中有一個警語即如果讀了一個paddle，則不能太快讀取另一個paddle。下面爲一機器語言程式，其中paddles之間的干擾能夠被表露證實（ demonstrated ）。

打入監督程式命令xxxxY以起動這個程式，其中xxxx是一個數值用來代表在讀paddleϕ和讀paddle 1之間，所使用的延遲時間，且Y代表控制Y。監督程式的命令"控制Y"造成一個JMP到位置$ϕ3F8，在那裡我們擺一個JMP到此程式的開始處。

當監督程式掃瞄此輸入命令列，16進位數元的值被放在第零頁位置A1L,H（$3C-3D）以供我們使用。

Paddle的干擾—程式範例

```
03F8  JMP  $2000

2000  LDA  $#C0     Set counter for 64 samples to run
2002  STA  $4       before clearing screen and starting over.
2004  LDA  $3C      Pick up low part of entered count from AIL
2006  STA  $10      and store it for repeated use.
2008  LDA  $3D      Pick up high part of entered count from AIH
200A  STA  $11      and store it for repeated use.

200C  LDA  $10      Pick up low part of count:
200E  STA  $12      store it in counter for this pass,
2010  LDA  $11      and also high part.
2012  STA  $13
2014  LDX  $#0      Set X for paddle 0 read.
2016  JSR  $FB1E    Call paddle read.
2019  STY  $0       Store paddle 0 result in location 0.

201B  DEC  $12      Count down delay loop low byte:
201D  BNE  $201B    when zero, count down high byte.
201F  DEC  $13
2021  BMI  $201B    Stay in the loop until high goes minus.

2023  LDX  $#1      Set X for paddle 1 read.
2025  JSR  $FB1E    Call paddle read.
2028  STY  $1       Store paddle 1 result in location 1.

202A  LDA  $0       Pick up paddle 0 value.
202C  JSR  $FDDA    Print it as a hex value.
202F  LDA  $#A0     Pick up a blank to print.
2031  JSR  $FDED    Print the blank.
2034  LDA  $1       Pick up paddle 1 value.
2036  JSR  $FDDA    Print it as a hex value.
2039  JSR  $F948    Print three blanks.

203C  INC  $5       Delay for awhile to keep paddle 1 read
203E  BNE  $203C    from upsetting paddle 0 results.

2040  INC  $4       Is it time to clear screen and restart?
2042  BNE  $200C    NE means no, go back and sample again.

2044  LDA  $#0      Wait a while before clearing screen.
2046  STA  $4
2048  STA  $5
```

```
204A   INC   $4
204C   BNE   $204A
204E   INC   $5
2050   BNE   $204A
2052   JSR   $FC58    Clear the screen.
2055   LDA   $#C0     Restore the per screen counter,
2057   STA   $4
2059   BNE   $200C     and go one more big round.
```

暫存器在BASIC叫用監督程式時的用途
REGISTERS FOR BASIC MONITOR CALLS

在本書中所談的許多進入點需要先設定暫存器以便能正確地運作。下面爲一個針對APPLESOFT而寫的例子程式，它叫用了監督程式以便將十進位轉換成十六進位。

這動作的原理是在一個監督程式G命令，在跑至PCL,H所指的位置之前，暫存器的值先從SAVE區域取回。所以，把目的地的位址POKE到PCL,H內並POKE 所需暫存器的內容到XREG, YREG, 在監督程式GO命令處理器中的進入點可被用來傳遞暫存器到所選定的常規去。

十進位到十六進位的轉換
DECIMAL TO HEX CONVERSION

Applesoft程式範例
APPLESOFT SAMPLE PROGRAM

```
10   REM CONVERT DECIMAL INPUT TO HEX OUTPUT
100  INPUT "ENTER NUMBER ";A      Read the input.
110  IF A=99999 THEN END          Provide a way to end the program.
150  C% = A / 256                 Isolate the high byte.
200  POKE 71,C%                   Set YREG for PRNTYX call.
300  B% = A / 256                 Get remainder from A/256.
310  B = B% * 256                 For low byte (XREG) POKE.
320  B% = A - B
350  POKE 70,B%
400  POKE 59,249                  Set PCH to $F9.
500  POKE 58,64                   Set PCL to $40.
550  PRINT                        Print a blank line.
600  CALL 65209                   Entry point in GO processor is FF89
650  PRINT                        Print a blank line.
700  GOTO 100                     Go around for another number.
```

單步和追踪的特色
STEP AND TRACE PECULIARITIES

在舊型監督程式中的Step和Trace 功能在某些情況下。會不正確地顯示暫存器的內容。STEP常規對JSR, RTS , JMP ，間接JMP, RTI 和BRK 指令特別注意。在每種情況中，暫存器的內容是從位於$45-49 的SAVE 區域取出顯示。但是事實上在“執行”完這

些指令後，並沒有叫用SAVE，這是給一般的被追踪指令的作法，所以顯示出來的暫存器爲在執行這指令之前，存在SAVE區域的那些值。

因此，對於JSR和RTS ，所顯示的S暫存器內容是不對的。在JSR或RTS 後的第一個指令，除非它此亦爲一個RTS或JSR ，否則的話所顯示的S暫存器是不正確的。

此Step和Trace常規在處理一個BRK指令時是不對的。亦即，對 BRK 所顯示的位址是正確的，而不是下兩個位元組，因爲 BRK 是被STEP常規偵測而不是被65φ2執行的。

雖然Step和trace 在對某些程式的檢驗工作非常有用，但是它們不能用來追踪對監督程式的叫用（ calls to monitor ）（一般還包括"印 print "輸出），也不能追踪使用A1L,H到A4L,H的程式。

由於在PCADJ（$F953 ）缺少" CLD "，如果你在被追踪或單步化的程式裡頭設定十進位模式（SED），則將顯示不正確的位址。

附錄A 6502指令集

6502指令集

下列之符號將應用於本總結中：

A	：累加器
X，Y	：指標暫存器
M	：記憶器
C	：進位
P	：處理機狀態暫存器
S	：堆疊指示器
✓	：改變
—	：未改變
+	：加
∧	：邏輯AND
−	：減
∀	：邏輯互斥 OR
↑	：從堆疊中取出
↓	：堆至堆疊上
→	：傳至
←	：傳至
∨	：邏輯 OR
PC	：程式計數器

圖1　ASL：向左移一位元之動作

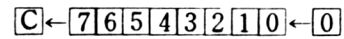


圖2　ROL：向左旋轉一位元（記憶器或累加器）

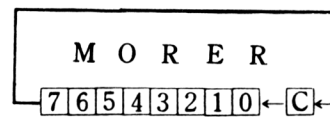
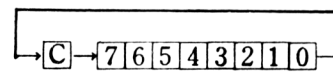


圖3　ROR：向右旋轉一位元



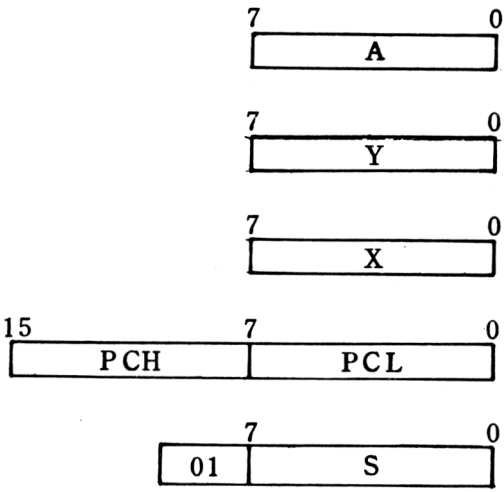
注

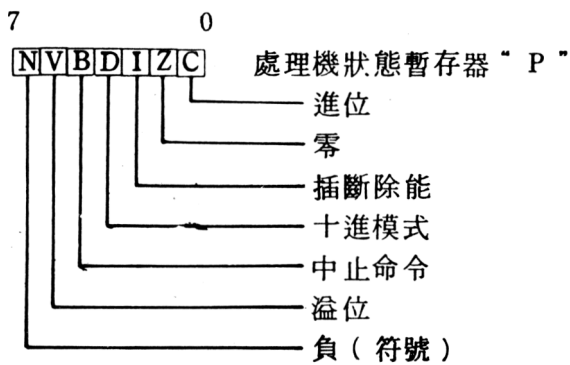
注意1：BIT：測試各位元，位元6及7被傳送至狀態暫存器，若A

PCH　：程式計數器之高位元組
PCL　：程式計數器之低位元組
OPER：運算子
＃　　：立即定址型式

∧M之結果爲0，則Z＝1，否則Z＝0。

規劃模式





ADC　—與進位值一起加至累加器—

NZCIDV
√√√--√

運算：A＋M＋C→A，C

定址法	組合語言形式	運算碼	位元組數	週期數
立即	ADC　#dd	69	2	2
零頁	ADC　aa	65	2	3
零頁，X	ADC　aa，X	75	2	4
絕對	ADC　aaaa	6D	3	4
絕對，X	ADC　aaaa，X	7D	3	4*
絕對，Y	ADC　aaaa，Y	79	3	4*
（間接，X）	ADC　（aa，X）	61	2	6
（間接），Y	ADC　（aa），Y	71	2	5*

* 若超出頁界限，則加1。

AND　—記憶位置內含與累加器內含作AND運算—

NZCIDV
√√----

運算：A∧M→A

定址法	組合語言形式	運算碼	位元組數	週期數
立即	AND　#dd	29	2	2
零頁	AND　aa	25	2	3
零頁，X	AND　aa，X	35	2	4
絕對	AND　aaaa	2D	3	4
絕對，X	AND　aaaa，X	3D	3	4*
絕對，Y	AND　aaaa，Y	39	3	4*
（間接，X）	AND　（aa，X）	21	2	6
（間接），Y	AND　（aa），Y	31	2	5*

* 超越頁界限時加1。

ASL　　—累加器內含左移—

運算：C ← 7 6 5 4 3 2 1 0 ← 0

NZCIDV
√√√---

定址法	組合語言形式		運算碼	位元組數	週期數
累加器	ASL	A	0A	1	2
零頁	ASL	aa	06	2	5
零頁，X	ASL	aa,X	16	2	6
絕對	ASL	aaaa	0E	3	6
絕對，X	ASL	aaaa,X	1E	3	7

BCC　　—進位清除時跳越—

AZCIDV

運算：c = 0 時跳越

定址法	組合語言形式		運算碼	位元組數	週期數
相對	BCC	aa	90	2	2*

* 跳越至同頁時加 1，跳至不同頁時加 2。

附註：AIM65 能接受 BCC　aaaa　之絕對定址形式，並將之轉換成相對位址。

BCS　　—進位置定時跳越—

NZCIDV

運算：c = 1 時跳越。

定址法	組合語言形式		運算碼	位元組數	週期數
相對	BCS	aa	BO	2	2*

* 跳越至同頁時加 1，跳至次頁時加 2。

註：AIM65 可接受絕對定址之運算元，並將之換成相對位址。

BEQ　　—結果等於零時跳越—

NZCIDV

運算：Z = 1 時跳越發生

定址法	組合語言形式	運算碼	位元組數	週期數
相對	B E Q　　a a	F0	2	2*

* 跳至同頁時加 1 ，跳至次一頁時加 2 。

註：AIM65 可接受絕對位址之運算元（形如 BEQ　aaaa），並將之轉換成相對位址 。

BIT　　—測試記憶位元值—

運算：A∧M，M_7 → N，M_6 → V

第 6 與 7 位元抄至狀態暫存器 。　　NZCIDV

若 A∧M 結果爲零，則 Z = 1 ；否則 Z = 0 。　　M_7 √ − − − M_6

定址法	組合語言形式	運算碼	位元組數	週期數
零頁	B I T　　a a	24	2	3
絕對	B I T　　a a a a	2C	3	4

BMI　　—運算結果負時跳越—

NZCIDV

運算：N = 1時，跳越發生 。

定址法	組合語言形式	運算碼	位元組數	週期數
相對	BMI　　a a	30	2	2*

* 跳至同頁時加 1 ，跳至不同頁時加 2 。

註：A I M 65 能接受絕對位址，並將之轉換成一相對位址 。

BNE —結果不等於零時跳越—

NZCIDV

運算：Z = 0時，跳越發生。

定址法	組合語言形式	運算碼	位元組數	週期數
相對	BNE aa	D0	2	2*

* 跳越至同頁時加1，跳越至不同頁時加2。

註：AIM 65 能接受一絕對位址，並將之轉換成相對位址。

BPL —結果爲正時跳越—

NZCIDV

運算：N = 0時，跳越發生。

定址法	組合語言形式	運算碼	位元組數	週期數
相對	BPL aa	10	2	2*

* 跳越至同頁時加1，跳越至不同頁時加2。

註：AIM 65 能接受一絕對位址之運算元（指令格式BPL aaaa），並將之轉換成相對位址。

BRK —迫使插斷—

BNZCIDV
1---1--

運算：被迫插斷PC+2↓P↓

定址法	組合語言形式	運算碼	位元組數	週期數
隱含	BRK	00	1	7

BVC　　—溢位清除時跳越—

NZCIDV

運算：V = 0 時，跳越發生。

定址法	組合語言形式	運算碼	位元組數	週期數
相對	BVC　　aa	50	2	2*

* 跳越至同頁時加 1，跳越至不同頁時加 2。

註：AIM 65 能接受一絕對位址，並將之轉換成相對位址。

BVS　　—溢位旗號置定時跳越—

NZCIDV

運算：V = 1 時，跳越發生。

定址法	組合語言形式	運算碼	位元組數	週期數
相對	BVS　　aa	70	2	2*

* 跳越至同頁時加 1，跳越至不同頁時加 2。

註：AIM 65 能接受一絕對位址，並將之轉換成相對位址。

CLC　　—清除進位旗號—

NZCIDV
--0---

運算：0 → C

定址法	組合語言形式	運算碼	位元組數	週期數
隱含	CLC	18	1	2

CLD —清除十進制位元—

NZCIDV
-----0-

運算：0 → D

定址法	組合語言形式	運算碼	位元組數	週期數
隱含	CLD	D8	1	2

CLI —清除插斷禁能位元—

NZCIDV
---0--

運算：0 → I

定址法	組合語言形式	運算碼	位元組數	週期數
隱含	CLI	58	1	2

CLV —清除溢位旗號—

NZCIDV
------0

運算：0 → V

定址法	組合語言形式	運算碼	位元組數	週期數
隱含	CLV	B8	1	2

CMP　　－記憶位置內含與累加器內含相比－

NZCIDV
√√√－－－

運算：A－M

定址法	組合語言形式	運算碼	位元組數	週期數
立即	CMP　#dd	C9	2	2
零頁	CMP　aa	C5	2	3
零頁，X	CMP　aa，X	D5	2	4
絕對	CMP　aaaa	CD	3	4
絕對，X	CMP　aaaa，X	DD	3	4*
絕對，Y	CMP　aaaa，Y	D9	3	4*
(間接，X)	CMP　(aa，X)	C1	2	6
(間接)，Y	CMP　(aa)，Y	D1	2	5*

＊若越過頁邊界，則加1。

CPX　　－記憶位置內含與X暫存器內含相比－

NZCIDV
√√√－－－

運算：X－M

定址法	組合語言形式	運算碼	位元組數	週期數
立即	CPX　#dd	E0	2	2
零頁	CPX　aa	E4	2	3
絕對	CPX　aaaa	EC	3	4

CPY　　－記憶位置內含與Y暫存器內含相比

NZCIDV
√√√－－－

運算：Y－M

定址法	組合語言形式	運算碼	位元組數	週期數
立即	CPY　#dd	C0	2	2
零頁	CPY　aa	C4	2	3
絕對	CPY　aaaa	CC	3	4

DEC

－記憶位置內含值減1－

NZCIDV
√√----

運算：M－1→M

定址法	組合語言形式	運算碼	位元組數	週期數
零頁	DEC aa	C6	2	5
零頁，X	DEC aa，X	D6	2	6
絕對	DEC aaaa	CE	3	6
絕對，X	DEC aaaa，X	DE	3	7

DEX

－X暫存器內含值減1－

NZCIDV
√√----

運算：X－1→X

定址法	組合語言形式	運算碼	位元組數	週期數
隱含	DEX	CA	1	2

DEY

－Y暫存器內含值減1－

NZCIDV
√√----

運算：Y－1→Y

定址法	語言組合形式	運算碼	位元組數	週期數
隱含	DEY	88	1	2

EOR　—記憶位置值與累加器值作EOR運算—

NZCIDV
√√----

運算：AVM→A

定址法	組合語言形式	運算碼	位元組數	週期數
立即	EOR #dd	49	2	2
零頁	EOR aa	45	2	3
零頁，X	EOR aa,x	55	2	4
絕對	EOR aaaa	4D	3	4
絕對，X	EOR aaaa,X	5D	3	4*
絕對，Y	EOR aaaa,Y	59	3	4*
(間接，X)	EOR (aa,X)	41	2	6
(間接)，Y	EOR (aa),Y	51	2	5*

INC　—記憶位置內含值加1—

NZCIDV
√√----

運算：M + 1→M

定址法	組合語言形式	運算碼	位元組數	週期數
零頁	INC aa	E6	2	5
零頁，X	INC aa,X	F6	2	6
絕對	INC aaaa	EE	3	6
絕對，X	INC aaaa,X	FE	3	7

INX　—X暫存器內含值加1—

NZCIDV
√√----

運算：X + 1→X

定址法	組合語言形式	運算碼	位元組數	週期數
隱含	INX	E8	1	2

INY　　—Y暫存器內含值加1—

NZCIDV
√√----

運算：Y＋1→Y

定址法	組合語言形式	運算碼	位元組數	週期數
隱含	INY	C8	1	2

JMP　　—跳越—

運算：(PC+1)→PCL
　　(PC+2)→PCH

NZCIDV

定址法	組合語言形式	運算碼	位元組數	週期數
絕對	JMP　aaaa	4C	3	3
間接	JMP　(aaaa)	6C	3	5

JSR　　—跳越至副程式—

運算：PC+2↓,(PC+1)→PCL
　　(PC+2)→PCH

NZCIDV

定址法	組合語言形式	運算碼	位元組數	週期數
絕對	JSR　aaaa	20	3	6

LDA　　—把記憶位置內含放入累加器—

NZCIDV
√√----

運算：M→A

定址法	組合語言形式	運算碼	位元組數	週期數
立即	LDA　#dd	A9	2	2
零頁	LDA　aa	A5	2	3
零頁，X	LDA　aa,X	B5	2	4
絕對	LDA　aaaa	AD	3	4
絕對，X	LDA　aaaa,X	BD	3	4*
絕對，Y	LDA　aaaa,Y	B9	3	4*
(間接，X)	LDA　(aa,X)	A1	2	6
(間接),Y	LDA　(aa),Y	B1	2	5*

* 若超越頁邊界，則加1。

LDX　　—把記憶位置內含放入X暫存器—

NZCIDV
√√----

運算：M→X

定址法	組合語言形式	運算碼	位元組數	週期數
立即	LDX　#dd	A2	2	2
零頁	LDX　aa	A6	2	3
零頁，Y	LDX　aa,Y	B6	2	4
絕對	LDX　aaaa	AE	3	4
絕對，Y	LDX　aaaa,Y	BE	3	4*

* 若越過頁邊界，則加1。

LDY —記憶位置內含值放入Y暫存器—

NZC!DV
√√----

運算：M→Y

定址法	組合語言形式	運算碼	位元組數	週期數
立即	LDY #dd	A0	2	2
零頁	LDY aa	A4	2	3
零頁，X	LDY aa,X	B4	2	4
絕對	LDY aaaa	AC	3	4
絕對，X	LDY aaaa,X	BC	3	4*

*越過頁界限時加1。

LSR —邏輯右移—

NZCIDV
0√√---

運算：0→[7][6][5][4][3][2][1][0]→C

定址法	組合語言形式	運算碼	位元組數	週期數
累加器	LSR A	4A	1	2
零頁	LSR aa	46	2	5
零頁，X	LSR aa,X	56	2	6
絕對	LSR aaaa	4E	3	6
絕對，X	LSR aaaa,X	5E	3	7

NOP —無運算—

運算：無運算

NZCIDV

定址法	組合語言形式	運算碼	位元組數	週期數
隱含	NOP	EA	1	2

ORA　　—記憶位置內含與累加器內作OR運算—

NZCIDV
√√----

運算：A V M→A

定址法	組合語言形式	運算碼	位元組數	週期數
立即	ORA #dd	09	2	2
零頁	ORA aa	05	2	3
零頁，X	ORA aa,X	15	2	4
絕對	ORA aaaa	0D	3	4
絕對，X	ORA aaaa,X	1D	3	4*
絕對，Y	ORA aaaa,Y	19	3	4*
(間接，X)	ORA (aa,X)	01	2	6
(間接)，Y	ORA (aa),Y	11	2	5*

*超越頁界時加1。

PHA　　—累加器內含推入堆疊器—

NZCIDV

運算：A ↓

定址法	組合語言形式	運算碼	位元組數	週期數
隱含	PHA	48	1	3

PHP　　—狀態暫存器內含推入堆疊器—

NZCIDV

運算：P ↓

定址法	組合語言形式	運算碼	位元組數	週期數
隱含	PHP	08	1	3

PLA

—自堆疊器拉取至累加器—

NZCIDV

運算：A ↑ √√----

定址法	組合語言形式	運算碼	位元組數	週期數
隱含	PLA	68	1	4

PLP

—狀態暫存器內含自堆疊器拉回

NZCIDV

運算：P ↑ 來自堆疊內含

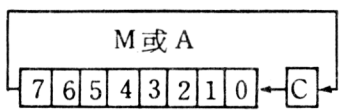
定址法	組合語言形式	運算碼	位元組數	週期數
隱含	PLP	28	1	4

ROL

—左旋轉—

NZCIDV

運算：



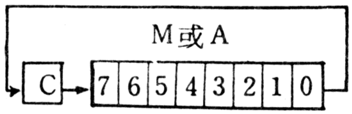
√√√---

定址法	組合語言形式	運算碼	位元組數	週期數
累加器	ROL A	2A	1	2
零頁	ROL aa	26	2	5
零頁，X	ROL aa,X	36	2	6
絕對	ROL aaaa	2E	3	6
絕對，X	ROL aaaa,X	3E	3	7

ROR　　—右旋轉—

NZCIDV

√√√---

運算：


定址法	組合語言形式	運算碼	位元組數	週期數
累加器	ROR　A	6A	1	2
零頁	ROR　aa	66	2	5
零頁，X	ROR　aa,X	76	2	6
絕對	ROR　aaaa	6E	3	6
絕對，X	ROR　aaaa,X	7E	3	7

RTI　　—插斷回返—

NZCIDV

運算：P ↑ PC ↑　　來自堆疊內含

定址法	組合語言形式	運算碼	位元組數	週期數
隱含	RTI	40	1	6

RTS　　—副程式回返—

NZCIDV

運算：PC↑，PC+1→PC

定址法	組合語言形式	運算碼	位元組數	週期數
隱含	RTS	60	1	6

SBC —累加器值減去記憶位置值與進位—

運算：A－M－C →A

NZCIDV
√√v−−√

附註：C＝借位

定址法	組合語言形式	運算碼	位元組數	週期數
立即	SBC #dd	E9	2	2
零頁	SBC aa	E5	2	3
零頁，X	SBC aa,X	F5	2	4
絕對	SBC aaaa	ED	3	4
絕對，X	SBC aaaa,X	FD	3	4*
絕對，Y	SBC aaaa'Y	F9	3	4*
(間接，X)	SBC (aa,X)	E1	2	6
(間接)，Y	SBC (aa),Y	F1	2	5*

* 若超越頁界限，則加1。

SEC —進位旗號置定爲1—

NZCIDV
--1---

運算：1→C

定址法	組合語言形式	運算碼	位元組數	週期數
隱含	SEC	38	1	2

SED —置定爲十進制—

NZCIDV
----1-

運算：1→D

定址法	組合語言形式	運算碼	位元組數	週期數
隱含	SED	F8	1	2

SEI　—置定插斷禁能狀態—

NZCIDV

運算：1→I　　---1--

定址法	組合語言形式	運算碼	位元組數	週期數
隱含	SEI	78	1	2

STA　—累加器內含存出至記憶器—

NZCIDV

運算：A→M　　-----

定址法	組合語言形式	運算碼	位元組數	週期數
零頁	STA aa	85	2	3
零頁，X	STA aa,X	95	2	4
絕對	STA aaaa	8D	3	4
絕對，X	STA aaaa,X	9D	3	5
絕對，Y	STA aaaa,Y	99	3	5
(間接，X)	STA (aa,X)	81	2	6
(間接)，Y	STA (aa),Y	91	2	6

STX　—X暫存器內含存出記憶器—

NZCIDV

運算：X→M　　------

定址法	組合語言形式	運算碼	位元組數	週期數
零頁	STX aa	86	2	3
零頁，Y	STX aa,Y	96	2	4
絕對	STX aaaa	8E	3	4

STY　　—Y 暫存器內含存入記憶器—

NZCIDV

運算：Y→M

定址法	組合語言形式	運算碼	位元組數	週期數
零頁	STY　aa	84	2	3
零頁，X	STY　aa，X	94	2	4
絕對	STY　aaaa	8C	3	4

TAX　　—累加器內含抄至X暫存器—

NZCIDV
√√----

運算：A→X

定址法	組合語言形式	運算碼	位元組數	週期數
隱含	TAX	AA	1	2

TAY　　—累加器內含抄至Y暫存器—

NZCIDV
√√----

運算：A→Y

定址法	組合語言形式	運算碼	位元組數	週期數
隱含	TAY	A8	1	2

TSX　　－堆疊指示器內含抄至X暫存器－

NZCIDV

運算：S→X　　√√－－－－

定址法	組合語言形式	運算碼	位元組數	週期數
隱含	TSX	BA	1	2

TXA　　－X暫存器內含抄至累加器－

NZCIDV

運算：X→A　　√√－－－－

定址法	組合語言形式	運算碼	位元組數	週期數
隱含	TXA	8A	1	2

TXS　　－X暫存器內含抄至堆疊指示器－

NZCIDV

運算：X→S　　－－－－－－

定址法	組合語言形式	運算碼	位元組數	週期數
隱含	TXS	9A	1	2

TYA　　－Y暫存器內含抄至累加器－

NZCIDV

運算：Y→A　　√√－－－－

定址法	組合語言形式	運算碼	位元組數	週期數
隱含	TYA	98	1	2

附錄B 特殊位置

表1：鍵盤特殊位置

位置		描述
十六進制	十進制	
$C000	49152 -16384	鍵盤資料
$C010	49168 -16368	清除鍵盤激勵

表4：視頻顯示器記憶器範圍

顯示幕	頁	開始		結束	
		十六進制	十進制	十六進制	十進制
正文/低解析度	第一頁	$400	1024	$7FF	2047
	第二頁	$800	2048	$BFF	3071
高解析度	第一頁	$2000	8192	$3FFF	16383
	第二頁	$4000	16384	$5FFF	24575

表 5 ：顯示幕軟性開關

位置		描述
十六進制	十進制	
$C050	49232 -16304	顯示一種繪圖模式
$C051	49233 -16303	顯示正文模式
$C052	49234 -16302	顯示所有正文或繪圖模式
$C053	49235 -16301	正文與繪圖模式混合
$C054	49236 -16300	顯示第一頁
$C055	49237 -16299	顯示第二頁
$C056	49238 -16298	顯示低解析度繪圖模式
$C057	49239 -16297	顯示高解析度繪圖模式

表 9 ：告示特殊位置

告示	狀態	位置	
		十進制	十六進制
0	" 0 "	49240 -16296	$C058
	" 1 "	49241 -16295	$C059
1	" 0 "	49242 -16294	$C05A
	" 1 "	49243 -16293	$C05B
2	" 0 "	49244 -16292	$C05C
	" 1 "	49245 -16291	$C05D
3	" 0 "	49246 -16290	$C05E
	" 1 "	49247 -16289	$C05F

表 10 輸入／出特殊位置

功能	位置		讀／寫
	十進制	十六進制	
揚聲器	49200 -16336	$C030	讀
卡帶輸出	49184 -16352	$C020	讀
卡帶輸入	49256 -16288	$C060	讀
告示輸出	49240 至 49247	-16296 至 -16289	$C058 至 $C05F	讀／寫
旗標輸入	49249	-16287	$C061	讀
	49250	-16286	$C062	讀
	49251	-16285	$C063	讀
類比輸入	49252	-16284	$C064	讀
	49253	-16283	$C065	
	49254	-16282	$C066	
	49255	-16281	$C067	
類比清除	49264	-16272	$C070	讀／寫
實用激勵	49216	-16320	$C040	讀

表 11：正文窗特殊位置

功能	位置		最小／正常／最大值	
	十進制	十六進制	十進制	十六進制
左緣	32	$20	0／0／39	$0／$0／$17
寬	33	$21	0／40／40	$0／$28／$28
上緣	34	$22	0／0／24	$0／$0／$18
下緣	35	$23	0／24／24	$0／$18／$18

表12：正常/反相控制值

值		效果
十進制	十六進制	
255	$FF	COUT 將依正常模式顯示字元
63	$3F	COUT 將依反相模式顯示字元
127	$7F	COUT 將依閃爍模式顯示字母而依反相模式顯示其他字元

表13：自動啓始ROM特殊位置

位置		內含
十進制	十六進制	
1ø1ø 1ø11	$3F2 $3F3	軟性進口向量，該兩位置包含任何語言的進口點位址，一般爲$E003
1ø12	$3F4	開機位元組，一般爲$45
64367 (-1169)	$FB6F	該處爲機器語言設定開機位元組的位置

表 14 ：第三頁監督程式位置

位址		使用	
十進制	十六進制	監督程式 ROM	自動啓始 ROM
1Ø Ø 8 1Ø Ø 9	$3FØ $3F1	無	存放處理機器語言“BRK”要求副程式的位址，一般爲$FA59
1Ø1Ø 1Ø11	$3F2 $3F3	無	軟性進口向量
1Ø12	$3F4	無	開機位元組
1Ø13 1Ø14 1Ø15	$3F5 $3F6 $3F7	存放處理Applesoft Ⅱ “&”命令之副程式的跳越指令，一般爲$4C $58 $FF	
1Ø16 1Ø17 1Ø18	$3F8 $3F9 $3FA	存放處理使用者(CTRL Y)命令副程式的跳越指令	
1Ø19 1Ø2Ø 1Ø21	$3FB $3FC $3FD	存放處理不可遮擋中斷副程式的跳越指令	
1Ø22 1Ø23	$3FE $3FF	存放處理中斷要求副程式的位址	

表22：內造式I／O位置

	$Ø	$1	$2	$3	$4	$5	$6	$7	$8	$9	$A	$B	$C	$D	$E	$F
$CØØØ	鍵盤資料輸入															
$CØ1Ø	清除鍵盤激勵															
$CØ2Ø	卡帶輸出連續反相（軟性開關）															
$CØ3Ø	揚聲器連續反相（軟性開關）															
$CØ4Ø	實用激勵															
$CØ5Ø	gr	tx	nomix		mix		pri		sec		lores		hires		anØ	an1	an2	an3
$CØ6Ø	cin		pbl		pb2		pb3		gcØ		gc1		gc2		gc3	重覆$C060～$C067		
$CØ7Ø	遊戲控制器激勵															

縮寫之意義：

gr　：設定繪圖模式
tx　：設定正文模式
nomix：設定全部正文或繪圖
mix　：正文及繪圖混合
pri　：顯示第一頁
sec　：顯示第二頁
lores：顯示低解析度繪圖
hires：顯示高解析度繪圖
an　：告示輸出
pb　：按扭輸入
gc　：遊戲控制器輸入
cin　：卡帶輸入

表 23 ：周邊擴充板 I / O位置

	$Ø	$1	$2	$3	$4	$5	$6	$7	$8	$9	$A	$B	$C	$D	$E	$F
$CØ8Ø	槽數 0 1 2 3 4 5 6 7 之輸入 / 輸出															
$CØ9Ø																
$CØAØ																
$CØBØ																
$CØCØ																
$CØDØ																
$CØEØ																
$CØFØ																

表 24 ：周邊擴充板PROM位置

	$ØØ	$1Ø	$2Ø	$3Ø	$4Ø	$5Ø	$6Ø	$7Ø	$8Ø	$9Ø	$AØ	$BØ	$CØ	$DØ	$EØ	$FØ
$C1ØØ $C2ØØ $C3ØØ $C4ØØ $C5ØØ $C6ØØ $C7ØØ	槽數 1 2 3 4 5 6 7 之PROM預留位址															

表 25 ： I / O位置基礎位置

基礎位置	槽數 Ø	1	2	3	4	5	6	7
$CØ8Ø	$CØ8Ø	$CØ9Ø	$CØAØ	$CØBØ	$CØCØ	$CØDØ	$CØEØ	$CØFØ
$CØ81	$CØ81	$CØ91	$CØA1	$CØB1	$CØC1	$CØD1	$CØE1	$CØF1
$CØ82	$CØ82	$CØ92	$CØA2	$CØB2	$CØC2	$CØD2	$CØE2	$CØF2
$CØ83	$CØ83	$CØ93	$CØA3	$CØB3	$CØC3	$CØC3	$CØE3	$CØF3
$CØ84	$CØ84	$CØ94	$CØA4	$CØB4	$CØC4	$CØD4	$CØE4	$CØF4
$CØ85	$CØ85	$CØ95	$CØA5	$CØB5	$CØC5	$CØD5	$CØE5	$CØF5
$CØ86	$CØ86	$CØ96	$CØA6	$CØB6	$CØC5	$CØD6	$CØE6	$CØF6
$CØ87	$CØ87	$CØ97	$CØA7m	$CØB7	$CØC7	$CØD7	$CØE7	$CØF7
$CØ88	$CØ88	$CØ98	$CØA8	$CØB8	$CØC8	$CØD8	$CØE8	$CØF8
$CØ98	$CØ98	$CØ99	$CØ9A	$CØB9	$CØC9	$CØD9	$CØE9	$CØF9
$CØ8A	$CØ8A	$CØ9A	$CØAA	$CØBA	$CØCA	$CØDA	$CØEA	$CØFA
$CØ8B	$CØ8B	$CØ9B	$CØAB	$ CBB	$CØCB	$CØDB	$CØEB	$CØFB
$CØ8C	$CØ8C	$CØ9C	$CØAC	$CØBC	$CØCC	$CØDC	$CØEC	$CØFC
$CØ8D	$CØ8D	$CØ9D	$CØAD	$CØBD	$CØCD	$CØDD	$CØED	$CØFD
$CØ8E	$CØ8E	$CØ9E	$CØAE	$CØBE	$CØCE	$CØDE	$CØEE	$CØFE
$CØ8F	$CØ8F	$CØ9F	$CØAF	$CØBF	$CØCF	$CØDF	$CØEF	$CØFF
	I / O位置							

表 26 ： I / O隨意書寫 RAM位置

基礎位址	槽數 1	2	3	4	5	6	7
$Ø478	$Ø479	$Ø47A	$Ø47B	$Ø47C	$Ø47D	$Ø47E	$Ø47F
$Ø4F8	$Ø4F9	$Ø4FA	$Ø4FB	$Ø4FC	$Ø4FD	$Ø4FE	4Ø4FF
$Ø578	$Ø579	$Ø57A	$Ø57B	$Ø57C	$Ø58D	$Ø57E	$Ø57F
$Ø5F8	$Ø5F9	$Ø5FA	$Ø5FB	$Ø5FC	$Ø5FD	$Ø5FE	$Ø5FF
$Ø678	$Ø679	$Ø67A	$Ø67B	$Ø67C	$Ø67D	$Ø67E	$Ø67F
$Ø6F8	$Ø6F9	$Ø6FA	$Ø6FB	$Ø6FC	$Ø6FD	$Ø6FE	$Ø6FF
$Ø778	$Ø779	$Ø77A	$Ø77B	$Ø77C	$Ø77D	$Ø77E	$Ø77F
$Ø7F8	$Ø7F9	$Ø7FA	$Ø7FB	$Ø7FC	$Ø8FD	$Ø7FE	$Ø7FF

附錄C ROM之表列

AUTOSTART ROM LISTING
MONITOR ROM LISTING

AUTOSTART ROM LISTING

```
0000                    2 ********************************
0000                    3 *
0000                    4 * APPLE II
0000                    5 * MONITOR II
0000                    6 *
0000                    7 * COPYRIGHT 1978 BY
0000                    8 * APPLE COMPUTER, INC
0000                    9 *
0000                   10 * ALL RIGHTS RESERVED
0000                   11 *
0000                   12 * STEVE WOZNIAK
0000                   13 *
0000                   14 ********************************
0000                   15 *
0000                   16 * MODIFIED NOV 1978
0000                   17 * BY JOHN A
0000                   18 *
0000                   19 ********************************
F800                   20          ORG $F800
F800                   21          OBJ $2000
F800                   22 ********************************
F800                   23 LOC0     EQU $00
F800                   24 LOC1     EQU $01
F800                   25 WNDLFT   EQU $20
F800                   26 WNDWDTH  EQU $21
F800                   27 WNDTOP   EQU $22
F800                   28 WNDBTM   EQU $23
F800                   29 CH       EQU $24
F800                   30 CV       EQU $25
F800                   31 GBASL    EQU $26
F800                   32 GBASH    EQU $27
F800                   33 BASL     EQU $28
F800                   34 BASH     EQU $29
F800                   35 BAS2L    EQU $2A
F800                   36 BAS2H    EQU $2B
F800                   37 H2       EQU $2C
F800                   38 LMNEM    EQU $2C
F800                   39 V2       EQU $2D
F800                   40 RMNEM    EQU $2D
F800                   41 MASK     EQU $2E
F800                   42 CHKSUM   EQU $2E
F800                   43 FORMAT   EQU $2E
F800                   44 LASTIN   EQU $2F
F800                   45 LENGTH   EQU $2F
F800                   46 SIGN     EQU $2F
F800                   47 COLOR    EQU $30
F800                   48 MODE     EQU $31
F800                   49 INVFLG   EQU $32
F800                   50 PROMPT   EQU $33
F800                   51 YSAV     EQU $34
F800                   52 YSAV1    EQU $35
F800                   53 CSWL     EQU $36
F800                   54 CSWH     EQU $37
F800                   55 KSWL     EQU $38
F800                   56 KSWH     EQU $39
F800                   57 PCL      EQU $3A
F800                   58 PCH      EQU $3B
F800                   59 A1L      EQU $3C
F800                   60 A1H      EQU $3D
F800                   61 A2L      EQU $3E
F800                   62 A2H      EQU $3F
F800                   63 A3L      EQU $40
F800                   64 A3H      EQU $41
F800                   65 A4L      EQU $42
F800                   66 A4H      EQU $43
F800                   67 A5L      EQU $44
F800                   68 A5H      EQU $45
```

```
F800                    69 ACC      EQU $45      NOTE OVERLAP WITH A5H'
F800                    70 XREG     EQU $46
F800                    71 YREG     EQU $47
F800                    72 STATUS   EQU $48
F800                    73 SPNT     EQU $49
F800                    74 RNDL     EQU $4E
F800                    75 RNDH     EQU $4F
F800                    76 PICK     EQU $95
F800                    77 IN       EQU $0200
F800                    78 BRKV     EQU $3F0    ; NEW VECTOR FOR BRK
F800                    79 SOFTEV   EQU $3F2    ; VECTOR FOR WARM START
F800                    80 PWREDUP  EQU $3F4    ; THIS MUST = EOR #$A5 OF SOFTEV+1
F800                    81 AMPERV   EQU $3F5    ; APPLESOFT & EXIT VECTOR
F800                    82 USRADR   EQU $03F8
F800                    83 NMI      EQU $03FB
F800                    84 IRQLOC   EQU $3FE
F800                    85 LINE1    EQU $400
F800                    86 MSLOT    EQU $07F8
F800                    87 IOADR    EQU $C000
F800                    88 KBD      EQU $C000
F800                    89 KBDSTRB  EQU $C010
F800                    90 TAPEOUT  EQU $C020
F800                    91 SPKR     EQU $C030
F800                    92 TXTCLR   EQU $C050
F800                    93 TXTSET   EQU $C051
F800                    94 MIXCLR   EQU $C052
F800                    95 MIXSET   EQU $C053
F800                    96 LOWSCR   EQU $C054
F800                    97 HISCR    EQU $C055
F800                    98 LORES    EQU $C056
F800                    99 HIRES    EQU $C057
F800                   100 SETANO   EQU $C058
F800                   101 CLRANO   EQU $C059
F800                   102 SETAN1   EQU $C05A
F800                   103 CLRAN1   EQU $C05B
F800                   104 SETAN2   EQU $C05C
F800                   105 CLRAN2   EQU $C05D
F800                   106 SETAN3   EQU $C05E
F800                   107 CLRAN3   EQU $C05F
F800                   108 TAPEIN   EQU $C060
F800                   109 PADDLO   EQU $C064
F800                   110 PTRIG    EQU $C070
F800                   111 CLRROM   EQU $CFFF
F800                   112 BASIC    EQU $E000  將(A)的LSB移入carry內。然後在下面中依此carry，
F800                   113 BASIC2   EQU $E003  決定mask內的值。carry＝1（奇數線）則mask＝Fφ。
F800                   114          PAGE       carry＝φ（偶數線），則mask＝φF。
F800  4A               115 PLOT     LSR A
F801  08               116          PHP
F802  20 47 F8         117          JSR GBASCALC     根據(A)算出GBASL.H的值。
F805  28               118          PLP              取回p-暫存器。
F806  49 0F            119          LDA #$0F         carry＝φ，則跳至RTMASK。
F808  90 02            120          BCC RTMASK
F80A  69 E0            121          ADC #$E0         φF＋Eφ＋carry（＝1）＝Fφ。
F80C  85 2E            122 RTMASK   STA MASK         (A)放入mask中。
F80E  B1 26            123 PLOT1    LDA (GBASL),Y    拿出要畫的點，及其同位元組內的另一點。
F810  45 30            124          EOR COLOR        由EOR此式與下式，決定出所要畫的點及其顏色。
F812  25 2E            125          AND MASK
F814  51 26            126          EOR (GBASL),Y    此式將所要畫的點放回，但另一點則還原。因爲
F816  91 26            127          STA (GBASL),Y    MASK遮掉了。又再做EOR，故原來的值還原。
F818  60               128          RTS 返回。
F819  20 00 F8         129 HLINE    JSR PLOT         畫出一點。
F81C  C4 2C            130 HLINE1   CPY H2            (Y)是否＝(H2)
F81E  B0 11            131          BCS RTS1         是則跳至 RTS1。
F820  C8               132          INY              (Y)＋1。
F821  20 0E F8         133          JSR PLOT1        叫用PLOT1，畫出一點。
F824  90 F6            134          BCC HLINE1       跳至HLINE1，以決定是否應該停止。
F826  69 01            135 VLINEZ   ADC #$01
F828  48               136 VLINE    PHA              貯放(A)於堆疊器中。
F829  20 00 F8         137          JSR PLOT         叫用PLOT，以畫出一點。
F82C  68               138          PLA              從堆疊器中還原(A)。
F82D  C5 2D            139          CMP V2           比較是否小於V2。
F82F  90 F5            140          BCC VLINEZ       是則跳至VLINEZ，以繼續畫下一點。
F831  60               141 RTS1     RTS 返回。       貯放回原來的地方。

叫用PLOT，以決定出對應於(A)的(GBASL.H)和mask值。同時畫出一點。

從(A)＋1＋carry開始畫。僅當第一次存carry時，carry才有效，後皆無carry。
```

```
F832  A0 2F       142 CLRSCR  LDY #$2F      清除整個螢幕48條橫線。
F834  D0 02       143         BNE CLRSC2    跳至CLRSC2
F836  A0 27       144 CLRTOP  LDY #$27      清除圖畫區域40條橫線。
F838  B4 2D       145 CLRSC2  STY V2         貯放到V2，以做爲垂直線的尾部。
F83A  A0 27       146         LDY #$27      寬度共40條縱線。從第39條清除到第零條。
F83C  A9 00       147 CLRSC3  LDA #$00       color放0。
F83E  85 30       148         STA COLOR     (Y)－1。
F840  20 2E FB    149         JSR VLINE
F843  88          150         DEY
F844  10 F6       151         BPL CLRSC3    不爲負數，即尚未清除至左邊，則跳CLRSC3。
F846  60          152         RTS           返回。
F847              153         PAGE
F847  48          154 GBASCALC PHA          這段程式在低解析度畫圖時用到，其目的是算出對於列(A)
F848  4A          155         LSR A         的記憶體位置，而將結果貯放在（GBASL.H），餘皆與
F849  29 03       156         AND #$03      BASCALC同。目前的(A)爲要畫的列除以２。請參閱
F84B  09 04       157         ORA #$04      BASCALC處。
F84D  85 27       158         STA GBASH
F84F  68          159         PLA
F850  29 18       160         AND #$18
F852  90 02       161         BCC GBCALC
F854  69 7F       162         ADC #$7F
F856  85 26       163 GBCALC  STA GBASL
F858  0A          164         ASL A
F859  0A          165         ASL A
F85A  05 26       166         ORA GBASL
F85C  85 26       167         STA GBASL
F85E  60          168         RTS
F85F  A5 30       169         LDA COLOR     color的值加３。
F861  18          170         CLC
F862  69 03       171         ADC #$03
F864  29 0F       172 SETCOL  AND #$0F      (A)的右半部爲要放入color的值。
F866  85 30       173         STA COLOR     先存入右半部。
F868  0A          174         ASL A         再將右半部移入左半部。
F869  0A          175         ASL A
F86A  0A          176         ASL A         讓color左、右兩邊的值相等。
F86B  0A          177         ASL A         貯回color內。
F86C  05 30       178         ORA COLOR     返回。
F86E  85 30       179         STA COLOR     將(A)列(Y)行的點取出，放在(A)內。邏輯右移(A)。
F870  60          180         RTS
F871  4A          181 SCRN    LSR A         保留P-暫存器，主要是保留carry的，其是從(A)內
F872  08          182         PHP           的LSB移入。
F873  20 47 FB    183         JSR GBASCALC
F876  B1 26       184         LDA (GBASL),Y     算出對應於(A)的（GBASL.H）。
F878  28          185         PLP               取出對應(A)列(Y)行的點。
F879  90 04       186 SCRN2   BCC RTMSKZ        取回P-暫存器。
F87B  4A          187         LSR A         如果carry設定的話，則保留左部份的nibble。
F87C  4A          188         LSR A         如果carry被清除，則保留右半部的nibble。
F87D  4A          189         LSR A
F87E  4A          190         LSR A
F87F  29 0F       191 RTMSKZ  AND #$0F
F881  60          192         RTS
F882              193         PAGE
F882  A6 3A       194 INSDS1  LDX PCL        (PCL) 放入(X)內。
F884  A4 3B       195         LDY PCH        (PCH) 放入(Y)內。
F886  20 96 FD    196         JSR PRYX2      叫用PRYXZ，以印出（PCL.H）的值和“－”。
F889  20 48 F9    197         JSR PRBLNK     叫用PRBLNK，以印出三個空白。
F88C  A1 3A       198 INSDS2  LDA (PCL,X)
F88E  A8          199         TAY            將運算子取出，如其右邊的nibble爲３，７　將運算子
F88F  4A          200         LSR A          取出，如其右邊的nibble爲３，７，B，F，，即位元
F890  90 09       201         BCC IEVEN      φ與位元１皆爲１，則發生錯誤。同時$-判斷是否爲$89
F892  6A          202         ROR A          ，如是亦是錯誤。
F893  B0 10       203         BCS ERR
F895  C9 A2       204         CMP #$A2       如爲$89，則爲錯誤。
F897  F0 0C       205         BEQ ERR
F899  29 87       206         AND #$87
F89B  4A          207 IEVEN   LSR A      取出FMT1，，對應的值，再判斷是否爲錯誤的機械語言。
F89C  AA          208         TAX
F89D  BD 62 F9    209         LDA FMT1,X
F8A0  20 79 FB    210         JSR SCRN2
F8A3  D0 04       211         BNE GETFMT
F8A5  A0 80       212 ERR     LDY #$80
F8A7  A9 00       213         LDA #$00
F8A9  AA          214 GETFMT  TAX
```

```
F8AA- BD A6 F9    215          LDA FMT2,X     取出顯示格式。
F8AD- 85 2E       216          STA FORMAT     貯放至FORMAT。
F8AF- 29 03       217          AND #$03       從FORMAT中，決定出長度。
F8B1- 85 2F       218          STA LENGTH     貯放至LENGTH。
F8B3- 98          219          TYA            將原來要被反組合的碼放回。
F8B4- 29 8F       220          AND #$8F
F8B6- AA          221          TAX
F8B7- 98          222          TYA
F8B8- A0 03       223          LDY #$03       如爲$8A,$9A,$AA,$BA,$CA,$EA,則跳至
F8BA- E0 8A       224          CPX #$8A       MNNDX3。 從此段中取出指令的對應碼，
F8BC- F0 0B       225          BEQ MNNDX3     以便在MNEML.H表中找出指令的寫法。
F8BE- 4A          226 MNNDX1   LSR A
F8BF- 90 08       227          BCC MNNDX3
F8C1- 4A          228          LSR A
F8C2- 4A          229 MNNDX2   LSR A
F8C3- 09 20       230          ORA #$20
F8C5- 88          231          DEY
F8C6- D0 FA       232          BNE MNNDX2     叫用INSDS1，以印出（PCL.H）、"—"，
F8C8- C8          233          INY            三個空白，及對應列指令的一個碼於(A)內。並
F8C9- 88          234 MNNDX3   DEY            找出適當的值，填入FORMAT及LENGTH。
F8CA- D0 F2       235          BNE MNNDX1
F8CC- 60          236          RTS
F8CD- FF FF FF    237          DFB $FF,$FF,$FF
F8D0              238          PAGE           取出（PCL.H）於(A)內。
F8D0- 20 82 F8    239 INSTDSP  JSR INSDS1     叫用PRBYTE印出(A)。
F8D3- 48          240          PHA
F8D4- B1 3A       241 PRNTOP   LDA (PCL),Y    印一個空白列輸出裝置。
F8D6- 20 DA FD    242          JSR PRBYTE
F8D9- A2 01       243          LDX #$01       判斷是否整個指令都印出，如
F8DB- 20 4A F9    244 PRNTBL   JSR PRBL2       否的話，則跳回PRNTOP。
F8DE- C4 2F       245          CPY LENGTH
F8E0- C8          246          INY            將機械語言與組合語言間顯示的空白填滿。
F8E1- 90 F1       247          BCC PRNTOP
F8E3- A2 03       248          LDX #$03       取出對應於指令的(A)碼。
F8E5- C0 04       249          CPY #$04
F8E7- 90 F2       250          BCC PRNTBL     (A)放到(Y)內。
F8E9- 68          251          PLA
F8EA- AB          252          TAY
F8EB- B9 C0 F9    253          LDA MNEML,Y    取出MNEML.R的值，放到LMNEM.RMNEM內。
F8EE- 85 2C       254          STA LMNEM
F8F0- B9 00 FA    255          LDA MNEMR,Y
F8F3- 85 2D       256          STA RMNEM
F8F5- A9 00       257 NXTCOL   LDA #$00       到267式爲止，從LMNEM.RMNEM中找出指令的英文字元。
F8F7- A0 05       258          LDY #$05
F8F9- 06 2D       259 PRMN2    ASL RMNEM      MNEML的位元3到位元7爲第一個字元，MNEML
F8FB- 26 2C       260          ROL LMNEM      的位元2到位元0，及MNEMR的位元7到MNEMR
F8FD- 2A          261          ROL A          的位元6爲第2個字元，MNEMR的位元5到位元1
F8FE- 88          262          DEY            爲第3個字元。
F8FF- D0 F8       263          BNE PRMN2
F901- 69 BF       264          ADC #$BF       這些字元都需加上$BF，才成爲對應的ASCII碼。
F903- 20 ED FD    265          JSR COUT
F906- CA          266          DEX
F907- D0 EC       267          BNE NXTCOL
F909- 20 4B F9    268          JSR PRBLNK     印出三個空白。
F90C- A4 2F       269          LDY LENGTH
F90E- A2 06       270          LDX #$06       從FMT2中得到的顯示格式。用
F910- E0 03       271 PRADR1   CPX #$03       CHAR1與CHAR2表間的對應
F912- F0 1C       272          BEQ PRADR5     找出各種運算元的印出格式。
F914- 06 2E       273 PRADR2   ASL FORMAT
F916- 90 0E       274          BCC PRADR3
F918- BD B3 F9    275          LDA CHAR1-1,X
F91B- 20 ED FD    276          JSR COUT
F91E- BD B9 F9    277          LDA CHAR2-1,X
F921- F0 03       278          BEQ PRADR3
F923- 20 ED FD    279          JSR COUT
F926- CA          280 PRADR3   DEX
F927- D0 E7       281          BNE PRADR1
F929- 60          282          RTS
F92A- 88          283 PRADR4   DEY
F92B- 30 E7       284          BMI PRADR2
F92D- 20 DA FD    285          JSR PRBYTE     如果指令爲3個位元組，則先印出第3個位元組，再印出第2
F930- A5 2E       286 PRADR5   LDA FORMAT     個位元組。
F932- C9 E8       287          CMP #$E8
```

```
F934- B1 3A      288          LDA (PCL),Y   取出運算元。
F936- 90 F2      289          BCC PRADR4
F938             290          PAGE
F938- 20 56 F9   291 RELADR   JSR PCADJ3
F93B- AA         292          TAX
F93C- E8         293          INX
F93D- D0 01      294          BNE PRNTYX
F93F- C8         295          INY
F940- 98         296 PRNTYX   TYA           印出Y，X暫存器的十六進位值。
F941- 20 DA FD   297 PRNTAX   JSR PRBYTE    印出A，X暫存器的十六進位值。
F944- 8A         298 PRNTX    TXA
F945- 4C DA FD   299          JMP PRBYTE
F948- A2 03      300 PRBLNK   LDX #$03      印出三個空白。
F94A- A9 A0      301 PRBL2    LDA #$A0      把空白的碼放到(A)。
F94C- 20 ED FD   302 PRBL3    JSR COUT      叫用COUT印出。
F94F- CA         303          DEX           判斷是否印到第3個空白。
F950- D0 F8      304          BNE PRBL2     不是則跳至PRBL2。
F952- 60         305          RTS           返回。
F953- 38         306 PCADJ    SEC           設定carry旗誌。
F954- A5 2F      307 PCADJ2   LDA LENGTH    (A)內含LENGTH，以表示反組合指令的長度。
F956- A4 3B      308 PCADJ3   LDY PCH       (Y)內爲（PCH）。
F958- AA         309          TAX           (A)移入(X)。
F959- 10 01      310          BPL PCADJ4
F95B- 88         311          DEY           如爲負數，則(Y)－1。此式與下列二式，則依正數時，
F95C- 65 3A      312 PCADJ4   ADC PCL       則直接相加，而負數時，則取二的補數後，再相加。
F95E- 90 01      313          BCC RTS2
F960- C8         314          INY
F961- 60         315 RTS2     RTS           如果carry＝1，則增加Y，否則返回。故返回時
F962- 04         316 FMT1     DFB $04       PCL＝A-暫存器，PCH＝Y-暫存器。
F963- 20         317          DFB $20       從此表中，找出各個指令的顯示格式。
F964- 54         318          DFB $54
F965- 30         319          DFB $30
F966- 0D         320          DFB $0D
F967- 80         321          DFB $80
F968- 04         322          DFB $04
F969- 90         323          DFB $90
F96A- 03         324          DFB $03
F96B- 22         325          DFB $22
F96C- 54         326          DFB $54
F96D- 33         327          DFB $33
F96E- 0D         328          DFB $0D
F96F- 80         329          DFB $80
F970- 04         330          DFB $04
F971- 90         331          DFB $90
F972- 04         332          DFB $04
F973- 20         333          DFB $20
F974- 54         334          DFB $54
F975- 33         335          DFB $33
F976- 0D         336          DFB $0D
F977- 80         337          DFB $80
F978- 04         338          DFB $04
F979- 90         339          DFB $90
F97A- 04         340          DFB $04
F97B- 20         341          DFB $20
F97C- 54         342          DFB $54
F97D- 3B         343          DFB $3B
F97E- 0D         344          DFB $0D
F97F- 80         345          DFB $80
F980- 04         346          DFB $04
F981- 90         347          DFB $90
F982- 00         348          DFB $00
F983- 22         349          DFB $22
F984- 44         350          DFB $44
F985- 33         351          DFB $33
F986- 0D         352          DFB $0D
F987- C8         353          DFB $C8
F988- 44         354          DFB $44
F989- 00         355          DFB $00
F98A- 11         356          DFB $11
F98B- 22         357          DFB $22
F98C- 44         358          DFB $44
F98D- 33         359          DFB $33
F98E- 0D         360          DFB $0D
```

```
F9BF: CB          361          DFB $CB
F9C0: 44          362          DFB $44
F9C1: A9          363          DFB $A9
F9C2: 01          364          DFB $01
F9C3: 22          365          DFB $22
F9C4: 44          366          DFB $44
F9C5: 33          367          DFB $33
F9C6: 0D          368          DFB $0D
F9C7: 80          369          DFB $80
F9C8: 04          370          DFB $04
F9C9: 90          371          DFB $90
F9CA: 01          372          DFB $01
F9CB: 22          373          DFB $22
F9CC: 44          374          DFB $44
F9CD: 33          375          DFB $33
F9CE: 0D          376          DFB $0D
F9CF: 80          377          DFB $80
F9D0: 04          378          DFB $04
F9D1: 90          379          DFB $90
F9D2: 26          380          DFB $26
F9D3: 31          381          DFB $31
F9D4: 87          382          DFB $87
F9D5: 9A          383          DFB $9A
F9D6: 00          384 FMT2     DFB $00     此爲顯示格式及長度決定處，參看215式。
F9D7: 21          385          DFB $21
F9D8: 81          386          DFB $81
F9D9: 82          387          DFB $82
F9DA: 00          388          DFB $00
F9DB: 00          389          DFB $00
F9DC: 59          390          DFB $59
F9DD: 4D          391          DFB $4D
F9DE: 91          392          DFB $91
F9DF: 92          393          DFB $92
F9E0: 86          394          DFB $86
F9E1: 4A          395          DFB $4A
F9E2: 85          396          DFB $85
F9E3: 9D          397          DFB $9D
F9E4: AC          398 CHAR1    DFB $AC     運算元的顯示格式中，所需的一些符號。
F9E5: A9          399          DFB $A9
F9E6: AC          400          DFB $AC
F9E7: A3          401          DFB $A3
F9E8: A8          402          DFB $A8
F9E9: A4          403          DFB $A4
F9EA: D9          404 CHAR2    DFB $D9
F9EB: 00          405          DFB $00
F9EC: D8          406          DFB $D8
F9ED: A4          407          DFB $A4
F9EE: A4          408          DFB $A4
F9EF: 00          409          DFB $00
F9F0: 1C          410 MNEML    DFB $1C     BRK
F9F1: 8A          411          DFB $8A     PHP
F9F2: 1C          412          DFB $1C     BPL
F9F3: 23          413          DFB $23     CLC
F9F4: 5D          414          DFB $5D     JSR
F9F5: 8B          415          DFB $8B     PLP
F9F6: 1B          416          DFB $1B     BMI
F9F7: A1          417          DFB $A1     SEC
F9F8: 9D          418          DFB $9D     RTI
F9F9: 8A          419          DFB $8A     PHA
F9FA: 1D          420          DFB $1D     BVC
F9FB: 23          421          DFB $23     CLI
F9FC: 9D          422          DFB $9D     RTS
F9FD: 8B          423          DFB $8B     PLA
F9FE: 1D          424          DFB $1D     BVS
F9FF: A1          425          DFB $A1     SEI
FA00: 00          426          DFB $00     ???
FA01: 29          427          DFB $29     DEY
FA02: 19          428          DFB $19     BCC
FA03: AE          429          DFB $AE     TYA
FA04: 69          430          DFB $69     LDY
FA05: A8          431          DFB $A8     TAY
FA06: 19          432          DFB $19     BCS
FA07: 23          433          DFB $23     CLV
```

```
F9D8  24          434          DFB  $24   CPY
F9D9  53          435          DFB  $53   INY
F9DA  1B          436          DFB  $1B   BNE
F9DB  23          437          DFB  $23   CLD
F9DC  24          438          DFB  $24   CPX
F9DD  53          439          DFB  $53   INX
F9DE  19          440          DFB  $19   BEQ
F9DF  A1          441          DFB  $A1   SED
F9E0  00          442          DFB  $00   ???
F9E1  1A          443          DFB  $1A   BIT
F9E2  5B          444          DFB  $5B   JMP
F9E3  5B          445          DFB  $5B   JMP
F9E4  A5          446          DFB  $A5   STY
F9E5  69          447          DFB  $69   LDY
F9E6  24          448          DFB  $24   CPY
F9E7  24          449          DFB  $24   CPX
F9E8  AE          450          DFB  $AE   TXA
F9E9  AE          451          DFB  $AE   TXS
F9EA  AB          452          DFB  $AB   TAX
F9EB  AD          453          DFB  $AD   TSX
F9EC  29          454          DFB  $29   DEX
F9ED  00          455          DFB  $00   ???
F9EE  7C          456          DFB  $7C   NOP
F9EF  00          457          DFB  $00   ???
F9F0  15          458          DFB  $15   ASL
F9F1  9C          459          DFB  $9C   ROL
F9F2  6D          460          DFB  $6D   LSR
F9F3  9C          461          DFB  $9C   ROR
F9F4  A5          462          DFB  $A5   STX
F9F5  69          463          DFB  $69   LDX
F9F6  29          464          DFB  $29   DEC
F9F7  53          465          DFB  $53   INC
F9F8  B4          466          DFB  $B4   ORA
F9F9  13          467          DFB  $13   AND
F9FA  34          468          DFB  $34   EOR
F9FB  11          469          DFB  $11   ADC
F9FC  A5          470          DFB  $A5   STA
F9FD  69          471          DFB  $69   CMP
F9FE  23          472          DFB  $23   SBC
F9FF  A0          473          DFB  $A0   LDA
FA00  D8          474 MNEMR    DFB  $D8
FA01  62          475          DFB  $62
FA02  5A          476          DFB  $5A
FA03  48          477          DFB  $48
FA04  26          478          DFB  $26
FA05  62          479          DFB  $62
FA06  94          480          DFB  $94
FA07  B8          481          DFB  $B8
FA08  54          482          DFB  $54
FA09  44          483          DFB  $44
FA0A  C8          484          DFB  $C8
FA0B  54          485          DFB  $54
FA0C  68          486          DFB  $68
FA0D  44          487          DFB  $44
FA0E  E8          488          DFB  $E8
FA0F  94          489          DFB  $94
FA10  00          490          DFB  $00
FA11  B4          491          DFB  $B4
FA12  08          492          DFB  $08
FA13  84          493          DFB  $84
FA14  74          494          DFB  $74
FA15  B4          495          DFB  $B4
FA16  28          496          DFB  $28
FA17  6E          497          DFB  $6E
FA18  74          498          DFB  $74
FA19  F4          499          DFB  $F4
FA1A  CC          500          DFB  $CC
FA1B  4A          501          DFB  $4A
FA1C  72          502          DFB  $72
FA1D  F2          503          DFB  $F2
FA1E  A4          504          DFB  $A4
FA1F  8A          505          DFB  $8A
FA20  00          506          DFB  $00
```

```
FA21  AA          507          DFB  $AA
FA22  A2          508          DFB  $A2
FA23  A2          509          DFB  $A2
FA24  74          510          DFB  $74
FA25  74          511          DFB  $74
FA26  74          512          DFB  $74
FA27  72          513          DFB  $72
FA28  44          514          DFB  $44
FA29  68          515          DFB  $68
FA2A  B2          516          DFB  $B2
FA2B  32          517          DFB  $32
FA2C  B2          518          DFB  $B2
FA2D  00          519          DFB  $00
FA2E  22          520          DFB  $22
FA2F  00          521          DFB  $00
FA30  1A          522          DFB  $1A
FA31  1A          523          DFB  $1A
FA32  26          524          DFB  $26
FA33  26          525          DFB  $26
FA34  72          526          DFB  $72
FA35  72          527          DFB  $72
FA36  88          528          DFB  $88
FA37  C8          529          DFB  $C8
FA38  C4          530          DFB  $C4
FA39  CA          531          DFB  $CA
FA3A  26          532          DFB  $26
FA3B  48          533          DFB  $48
FA3C  44          534          DFB  $44
FA3D  44          535          DFB  $44
FA3E  A2          536          DFB  $A2
FA3F  C8          537          DFB  $C8
FA40              538          PAGE
FA40  85 45       539 IRQ      STA  ACC      將(A)貯放在ACC內。
FA42  68          540          PLA           將P-暫存器從堆疊器內拿出又放回。
FA43  48          541          PHA
FA44  0A          542          ASL  A        向(A)左移三次，以檢查BREAK是由CPU的
FA45  0A          543          ASL  A        IRQ發生，亦由軟體指令的BRK指令發生。
FA46  0A          544          ASL  A
FA47  30 03       545          BMI  BREAK    如爲硬體，則經由（φ3FE，φ3FF）
FA49  6C FE 03    546          JMP  (IRQLOC) 間接跳，以執行IRQ程式。
FA4C  28          547 BREAK    PLP
FA4D  20 4C FF    548          JSR  SAV1     叫用SAV1以貯存A以外的所有暫存器。(A)在539式已貯放過。
FA50  68          549          PLA           將程式計數器從暫存器取回，存入到PCL,H內。
FA51  85 3A       550          STA  PCL
FA53  68          551          PLA
FA54  85 3B       552          STA  PCH      經由BRK的向量，做間接跳的動作，一般皆指向OLDBRK。
FA56  6C F0 03    553          JMP  (BRKV)   ;BRKV WRITTEN OVER BY DISK BOOT
FA59  20 82 F8    554 OLDBRK   JSR  INSDS1   叫用INSDS1，以印出（PCL,H）和"－"。
FA5C  20 DA FA    555          JSR  RGDSP1   叫用RGDSP1以顯示暫存器的內容。
FA5F  4C 65 FF    556          JMP  MON      跳至MON，以便讀下個命令。
FA62  D8          557 RESET    CLD           ; DO THIS FIRST THIS TIME  清除十進位模式。
FA63  20 84 FE    558          JSR  SETNORM  設定畫面爲黑底白字。
FA66  20 2F FB    559          JSR  INIT     設定畫面的格式及顯示資料來源。
FA69  20 93 FE    560          JSR  SETVID   設定螢幕爲輸出裝置。
FA6C  20 89 FE    561          JSR  SETKBD   設定鍵盤爲輸入裝置。
FA6F  AD 58 C0    562 INITAN   LDA  SETANO  ; ANO = TTL HI      下面爲清除GAME I/O
FA72  AD 5A C0    563          LDA  SETAN1  ; AN1 = TTL HI      硬體爲一預設狀態。
FA75  AD 5D C0    564          LDA  CLRAN2  ; AN2 = TTL LO
FA78  AD 5F C0    565          LDA  CLRAN3  ; AN3 = TTL LO
FA7B  AD FF CF    566          LDA  CLRROM  ; TURN OFF EXTNSN ROM  清除擴充ROM的開關。
FA7E  2C 10 C0    567          BIT  KBDSTRB ; CLEAR KEYBOARD  清除鍵盤的觸發信號。
FA81  D8          568 NEWMON   CLD           清除十進位模式。
FA82  20 3A FF    569          JSR  BELL    ; CAUSES DELAY IF KEY BOUNCES
FA85  AD F3 03    570          LDA  SOFTEV+1; IS RESET HI
FA88  49 A5       571          EOR  #$A5    ;A FUNNY COMPLEMENT OF THE
FA8A  CD F4 03    572          CMP  PWREDUP ; PWR UP BYTE ???
FA8D  D0 17       573          BNE  PWRUP   ; NO SO PWRUP
FA8F  AD F2 03    574          LDA  SOFTEV  ; YES SEE IF COLD START
FA92  D0 0F       575          BNE  NOFIX   ; HAS BEEN DONE YET?
FA94  A9 E0       576          LDA  #$E0    ; ??          （φ3F2,φ3F3）爲φFφφφ，故改成爲
FA96  CD F3 03    577          CMP  SOFTEV+1; ??          φEφφ3，而跳入BASIC。
FA99  D0 08       578          BNE  NOFIX   ; YES SO REENTER SYSTEM
FA9B  A0 03       579 FIXSEV   LDY  #3      ; NO SO POINT AT WARM START
```

檢查是否已啟過機，即（φ3F3）互斥或（φ3F4）的結果是否爲φA5，如不是，則跳至PWRUP。

判斷（φ3F2,φ3F3）是否爲φEφφφ，如不是，則跳至NOFIX，以執行一個間接跳。

```
FA9D  8C F2 03     580          STY SOFTEV  ; FOR NEXT RESET          執行間接跳，而跳至目前
FAA0  4C 00 E0     581          JMP BASIC   ; AND DO THE COLD START 的語言處理機，以離開
FAA3  6C F2 03     582 NOFIX    JMP (SOFTEV)   SOFT ENTRY VECTOR    RESET 的處理。
FAA6               583 *********************************
FAA6  20 60 FB     584 PWRUP    JSR APPLEII  叫用APPLEII，以清除螢幕和將APPLE  印在螢幕上。
FAA9               585 SETPG3   EQU *       ; SET PAGE 3 VECTORS  設定第三頁的向量。
FAA9  A2 05        586          LDX #5 (φ3Fφ) =59，(φ3F1) =FA，(φ3F2) =φφ，(φ3F3)
FAAB  BD FC FA     587 SETPLP   LDA PWRCON-1,X ; WITH CNTRL B ADRS  =φEφ，(φ3F4) =45。
FAAE  9D EF 03     588          STA BRKV-1,X ; OF CURRENT BASIC
FAB1  CA           589          DEX                          故φEφ與45做互斥或爲φA5，表示已覆過機。
FAB2  D0 F7        590          BNE SETPLP  先將 (LOCO，LOC1) 放入φC800，此時X因設定第三頁，
FAB4  A9 C8        591          LDA #$C8    ; LOAD HI SLOT +1          故爲零。
FAB6  86 00        592          STX LOCO    ; SETPG3 MUST RETURN X=0
FAB8  85 01        593          STA LOC1    ; SET PTR H
FABA  A0 07        594 SLOOP    LDY #7      ; Y IS BYTE PTR  7放到(Y)，以便檢查磁碟的ID。
FABC  C6 01        595          DEC LOC1  判斷是否從槽7檢查到槽1了，如果檢查過了，  則表示無碟
FABE  A5 01        596          LDA LOC1  機存在，故跳至FIXSEV。
FAC0  C9 C0        597          CMP #$C0    ; CAT LAST SLOT YET?
FAC2  F0 D7        598          BEQ FIXSEV  ; YES AND IT CANT BE A DISK
FAC4  8D F8 07     599          STA MSLOT  參看第一章中的週邊控制器工作區域中MSLOT的述。
FAC7  B1 00        600 NXTBYT   LDA (LOCO),Y  ; FETCH A SLOT BYTE
FAC9  D9 01 FB     601          CMP DISKID-1,Y ; IS IT A DISK ??
FACC  D0 EC        602          BNE SLOOP   ; NO SO NEXT SLOT DOWN
FACE  88           603          DEY
FACF  88           604          DEY         ; YES SO CHECK NEXT BYTE
FAD0  10 F5        605          BPL NXTBYT  ; UNTIL 4 CHECKED
FAD2  6C 00 00     606          JMP (LOCO)  檢查磁碟的ID，驗對順序爲3C，φ3，φφ，2φ，共四次，
FAD5  EA           607          NOP  成594式Y 放入了。如果四次皆符合，則經由 (LOCO) 間接跳。如果
FAD6  EA           608          NOP  檢查過程中，一發現有一不符合，則跳回SLOOP，以判斷是否檢查到
FAD7               609 * REGDSP MUST ORG $FAD7                槽φ了。
FAD7  20 BE FD     610 REGDSP   JSR CROUT 下段爲印出暫存器的內容。印出Carriage return。
FADA  A9 45        611 RGDSP1   LDA #$45  將φφ45位置，即暫存器貯放的起始位置，放入(A3L,H)內。
FADC  85 40        612          STA A3L
FADE  A9 00        613          LDA #$00
FAE0  85 41        614          STA A3H
FAE2  A2 FB        615          LDX #$FB FB即-5放到X內，此爲寫程式高段處，不僅可以在618式中，當
FAE4  A9 A0        616 RDSP1    LDA #$A0 做指標，亦可在625式中，當做計算器用。
FAE6  20 ED FD     617          JSR COUT  印出一個空白。
FAE9  BD 1E FA     618          LDA RTBL-251, 先後印出A,X,Y,P,S的字母。
FAEC  20 ED FD     619          JSR COUT
FAEF  A9 BD        620          LDA #$BD  印出 "=" 。
FAF1  20 ED FD     621          JSR COUT
FAF4               622 * LDA ACC+5,X  此式爲LDA ACC+5,X，即將要顯示的暫存器內容放入(A)內。
FAF4  B5 4A        623          DFB $B5,$4A
FAF6  20 DA FD     624          JSR PRBYTE 叫用PRBYTE印出。
FAF9  E8           625          INX  增加X值，檢查是否爲零，則表示已顯示所有的暫存器內容，否則跳
FAFA  30 E8        626          BMI RDSP1  至RDSP1繼續顯示。
FAFC  60           627          RTS  返回。
FAFD  59 FA        628 PWRCON   DW  OLDBRK  用來表示已開過機。
FAFF  00 E0 45     629          DFB $00,$E0,$45
FP02  20 FF 00
FB05  FF           630 DISKID   DFB $20,$FF,$00,$FF  磁碟機的ID。
FB06  03 FF 3C     631          DFB $03,$FF,$3C
FB09  C1 D0 D0     632 TITLE    DFB $C1,$D0,$D0 APP  $D3, $D0, $C,
FB0C  CC C5 A0     633          DFB $CC,$C5,$A0 LEb  $C3, $C1, A0,
FB0F  DD DB        634          DFB $DD,$DB ]〔       $B8, $B1,
FB11               635 XLTBL    EQU *
FB11  C4 C2 C1     636          DFB $C4,$C2,$C1 DBA
FB14  FF C3        637          DFB $FF,$C3 SFF C
FB16  FF FF FF     638          DFB $FF,$FF,$FF
FB19               639 *  MUST ORG $FB19
FB19  C1 D8 D9     640 RTBL     DFB $C1,$D8,$D9  暫存器的名稱，A,X,Y.
FB1C  D0 D3        641          DFB $D0,$D3  P,S。
FB1E  AD 70 C0     642 PREAD    LDA PTRIG  起動paddle計時器。
FB21               643          LST ON
FB21  A0 00        644          LDY #$00 (Y)設爲零。
FB23  EA           645          NOP
FB24  EA           646          NOP
FB25  BD 64 C0     647 PREAD2   LDA PADDL0,X  讀第(X)個paddle計時器。
FB28  10 04        648          BPL RTS2D  如爲正，則跳出返回。
FB2A  C8           649          INY  增加Y值。
FB2B  D0 F8        650          BNE PREAD2  不爲零，則跳至PREAD2。
FB2D  88           651          DEY  (Y)-1放到(Y)內。
```

```
FB2E: 60          652 RTS2D   RTS  返回。
FB2F: A9 00         2 INIT    LDA #$00  貯放ΦΦ於status中。
FB31: 85 48         3         STA STATUS
FB33: AD 56 C0      4         LDA LORES 清除高解析度圖表模式。
FB36: AD 54 C0      5         LDA LOWSCR 設定顯示資料是從主顯示區域來。
FB39: AD 51 C0      6 SETTXT  LDA TXTSET  設定文章模式。
FB3C: A9 00         7         LDA #$00  (A)放ΦΦ，以便貯放到WNDTOP。
FB3E: F0 0D         8         BEQ SETWND  跳至SETWND。
FB40: AD 50 C0      9 SETGR   LDA TXTCLR  設定圖表模式。
FB43: AD 53 C0     10         LDA MIXSET  設定圖形/文字混合模式。
FB46: 20 36 F8     11         JSR CLRTOP  叫用CLRTOP以清除圖表。
FB49: A9 14        12         LDA #$14  設定WNDTOP為S14。
FB4B: 85 22        13 SETWND  STA WNDTOP  將(A)值放入WNDTOP。
FB4D: A9 00        14         LDA #$00 將畫面左端設定為零。
FB4F: 85 20        15         STA WNDLFT
FB51: A9 28        16         LDA #$28  將畫面寬度設定為S28。
FB53: 85 21        17         STA WNDWDTH
FB55: A9 18        18         LDA #$18  將畫面底部設定為S18。
FB57: 85 23        19         STA WNDBTM
FB59: A9 17        20         LDA #$17  (CV)設定為S17。
FB5B: 85 25        21 TABV    STA CV
FB5D: 4C 22 FC     22         JMP VTAB  跳至VTAB；以算出（BASL H）的適當值。
FB60: 20 58 FC     23 APPLE!! JSR HOME   CLEAR THE SCRN  叫用HOME，以清除畫面。
FB63: A0 08        24         LDY #8  由於要印出的APPLE b )(共有八個字，故Y放8。
FB65: B9 08 FB     25 STITLE  LDA TITLE-1,Y   GET A CHAR  將APPLE b)( 從右邊逐一放入主
FB68: 99 GE 04     26         STA LINE1+14,Y顯示區域第一列的中，故APPLE b )(是從〔開始印出的。
FB6B: 86           27         DEY                    不相同的話，則表示無暫停要求。故僅當輸出CR時，
FB6C: D0 F7        28         BNE STITLE                才檢查是否有暫停要求。
FB6E: 60           29         RTS
FB6F: AD F3 03     30 SETPWRC LDA SOFTEV+1  此段表示已經睡過機了。即（ΦF3）互斥或（ΦF4）
FB72: 49 A5        31         EOR #$A5  的結果為ΦA5。
FB74: 8D F4 03     32         STA PWREDUP                檢查鍵盤暫存器，是否有鍵輸入。
FB77: 60           33         RTS
FB78               34 VIDWAIT EQU *      , CHECK FOR A PAUSE
FB78: C9 8D        35         CMP #$8D   , ONLY WHEN I HAVE A CR    (A)與carry return
FB7A: D0 18        36         BNE NOWAIT , NOT SO. DO REGULAR       (S80)比較，
FB7C: AC 00 C0     37         LDY KBD    , IS KEY PRESSED? 輸入鍵盤暫存器，檢查是否有鍵摩
FB7F: 10 13        38         BPL NOWAIT , NO 沒有的話，則跳至NOWAIT。          下去了。
FB81: C0 93        39         CPY #$93   , IS IT CTL S ?  檢查輸入的字元是否為控制-S。
FB83: D0 0F        40         BNE NOWAIT , NO SO IGNORE  不是的話，則跳至NOWAIT。
FB85: 2C 10 C0     41         BIT KBDSTRB , CLEAR STROBE 是的話，則清除掉鍵盤輸入觸發信號
FB88: AC 00 C0     42 KBDWAIT LDY KBD    , WAIT TILL NEXT KEY TO RESUME
FB8B: 10 FB        43         BPL KBDWAIT , WAIT FOR KEYPRESS 如無的話，則跳回上式。
FB8D: C0 83        44         CPY #$83   , IS IT CONTROL C ?檢查鍵入的字元是否為控制-C。
FB8F: F0 03        45         BEQ NOWAIT , YES SO LEAVE IT是的話，則跳至NOWAIT，而將控
FB91: 2C 10 C0     46         BIT KBDSTRB , CLR STROBE  制-C放入鍵盤暫存器內。故無執行下式。
FB94: 4C FD FB     47 NOWAIT  JMP VIDOUT , DO AS BEFORE  清除掉鍵盤輸入觸發信號。
FB97               48         PAGE                       跳至VIDOUT。
FB97: 38           49 ESCOLD  SEC          , INSURE CARRY SET 設定CARRY旗號。
FB98: 4C 2C FC     50         JMP ESC1     跳至ESC1以處理ESC鍵功能A，B，C，D，E，F。
FB9B: AA           51 ESCNOW  TAY          , USE CHAR AS INDEX 將(A)移到(Y)內。
FB9C: B9 48 FA     52         LDA XLTBL-$C9,Y , GET IJKM TO CBAD
FB9F: 20 97 FB     53         JSR ESCOLD , DO THIS CURSOR MOTION 叫用ESCOLD。
FBA2: 20 0C FD     54         JSR RDKEY  , AND GET NEXT
FBA5: C9 CE        55 ESCNEW  CMP #$CE   , IS THIS AN N ?
FBA7: B0 EE        56         BCS ESCOLD , N OR GREATER DO IT  如果大於或等於N則跳至
FBA9: C9 C9        57         CMP #$C9   , LESS THAN I ?(A)的字元與I比較。    ESCOLD。
FBAB: 90 EA        58         BCC ESCOLD , YES SO OLD WAY 如果小於I 則跳至ESCOLD。
FBAD: C9 CC        59         CMP #$CC   , IS IT A L ?  (A)的字元與L比較。
FBAF: F0 E6        60         BEQ ESCOLD , DO NORMAL  如果為L 則跳至ESCOLD。
FBB1: D0 EB        61         BNE ESCNOW , GO DO IT  不為L 則跳至ESCNOW。
FBB3: EA           62         NOP
FBB4: EA           63         NOP    將I，J，K，M轉換成為D，B，A，C。在$FB14的FF則
FBB5: EA           64         NOP    對應到沒用的L。
FBB6: EA           65         NOP
FBB7: EA           66         NOP    再讀下一個字元，故只要按一次ESC後，下個I，J，K，M仍
FBB8: EA           67         NOP    然有效。
FBB9: EA           68         NOP  (A)的字元與N比較。此處是在判斷(A)內是否為I，J，K，M，
FBBA: EA           69         NOP  如果不是的話則跳至ESCOLD。
```

```
FBBB: EA          70          NOP
FBBC: EA          71          NOP
FBBD: EA          72          NOP
FBBE: EA          73          NOP
FBBF: EA          74          NOP
FBC0: EA          75          NOP
FBC1:             76 *     MUST ORG $FBC1下面的位置一定爲ΦFBC1。BASCLAC必須
FBC1: 48          77 BASCALC PHA  保留A於堆疊器上。          參看第一章內的公式。
FBC2: 4A          78         LSR A (A)左移一位。故E 放到carry中。
FBC3: 29 03       79         AND #$03  將(A)形成000000CD。
FBC5: 09 04       80         ORA #$04  將(A)形成000001CD。
FBC7: 85 29       81         STA BASH  (A)放到BASH內。
FBC9: 68          82         PLA 從堆疊器上取回(A)體。
FBCA: 29 1B       83         AND #$1B  取出(A)中的AB。
FBCC: 90 02       84         BCC BASCLC2  如爲carry即E 不爲1，則跳至BASCLC2。
FBCE: 69 7F       85         ADC #$7F 加(A)+7F+carry（=1），故等於加上80。即
FBD0: 85 28       86 BASCLC2 STA BASL (A)放到（BASL）。          在(A)內填入E。
FBD2: 0A          87         ASL A  與下式將(A)變成EAB00000。
FBD3: 0A          88         ASL A
FBD4: 05 28       89         ORA BASL 將(A)變成EABAB000。
FBD6: 85 28       90         STA BASL(A)放回BASL。故算出對應於(A)的（BASL.H）。
FBD8: 60          91         RTS  返回。
FBD9: C9 87       92 BELL1   CMP #$87  判斷是否爲控制-G。
FBDB: D0 12       93         BNE RTS2B  是的則響蜘叭。不是的則返回。
FBDD: A9 40       94         LDA #$40  下面爲蜘叭以1KHZ響十分之一秒。
FBDF: 20 A8 FC    95         JSR WAIT  等0.01秒。
FBE2: A0 C0       96         LDY #$C0  Y=Sc中，則蜘叭響0.1秒。
FBE4: A9 0C       97 BELL2   LDA #$0C  1KHZ頻率。
FBE6: 20 A8 FC    98         JSR WAIT
FBE9: AD 30 C0    99         LDA SPKR  觸發蜘叭。
FBEC: 88         100         DEY
FBED: D0 F5      101         BNE BELL2
FBEF: 60         102 RTS2B   RTS     返回。
FBF0:            103         PAGE
FBF0: A4 24      104 STORADV LDY CH 將(CH)放到(Y)。
FBF2: 91 28      105         STA (BASL),Y  將輸出字元貯放到（BASL.H）+Y處。
FBF4: E6 24      106 ADVANCE INC CH  (CH)+1。
FBF6: A5 24      107         LDA CH  (CH)放到(A)內。
FBF8: C5 21      108         CMP WNDWDTH 比較(A)是否大於或等於WNDWDTH？
FBFA: B0 66      109         BCS CR     是則跳至CR。'
FBFC: 60         110 RTS3    RTS   返回。
FBFD: C9 A0      111 VIDOUT  CMP #$A0   判斷是否爲一般正常字元。
FBFF: B0 EF      112         BCS STORADV 是的話，則跳至STORADV。
FC01: AB         113         TAY  將(A)移至(Y)。
FC02: 10 EC      114         BPL STORADV 如果(A)小於$8Φ，表示爲閃爍字元或反相字元。
FC04: C9 8D      115         CMP #$8D                則跳至STORADV。
FC06: F0 5A      116         BEQ CR    判斷是否爲CR，是的話則跳至CR 常規。
FC08: C9 8A      117         CMP #$8A  判斷是否爲LF，是的話則跳至LF 常規。
FC0A: F0 5A      118         BEQ LF
FC0C: C9 88      119         CMP #$88  判斷是否爲Backspace。
FC0E: D0 C9      120         BNE BELL1  不是的話則跳至BELL1。
FC10: C6 24      121 BS      DEC CH  (CH)-1。
FC12: 10 E8      122         BPL RTS3 如爲正數，則跳至RTS3。
FC14: A5 21      123         LDA WNDWDTH    CH如不爲正數，則游標必須往上移。
FC16: 85 24      124         STA CH(A)放到(CH)。    （WNDWDTH）放到(A)。
FC18: C6 24      125         DEC CH （CH）-1，故(CH)移到右邊減1的位置。然後再
FC1A: A5 22      126 UP      LDA WNDTOP（WNDTOP）放到(A)。          進入上一列。
FC1C: C5 25      127         CMP CV  比較(A)與(CV)。
FC1E: B0 0B      128         BCS RTS4 如果（WNDTOP）=(CV)則返回，即不往上移。
FC20: C6 25      129         DEC CV  (CV)較大，故(CV)-1放回(CV)。
FC22: A5 25      130 VTAB    LDA CV  (CV)放到(A)內。      算出BASL.H對應於CV和
FC24: 20 C1 FB   131 VTABZ   JSR BASCALC                WNDLFT的記憶體位置。
FC27: 65 20      132         ADC WNDLFT 加上WNDLFT，以指向游標在捲動畫面左端的
FC29: 85 28      133         STA BASL 貯放(A)到BASL。          直正位置。
FC2B: 60         134 RTS4    RTS   返回。
FC2C: 49 C0      135 ESC1    EOR #$C0    ; ESC @ ?
FC2E: F0 28      136         BEQ HOME    ; IF SO DO HOME AND CLEAR
FC30: 69 FD      137         ADC #$FD    ; ESC-A OR B CHECK
FC32: 90 C0      138         LCC ADVANCE ; A, ADVANCE
FC34: F0 DA      139         BEQ BS      ; B, BACKSPACE
FC36: 69 FD      140         ADC #$FD    ; ESC-C OR D CHECK
FC38: 90 2C      141         BCC LF      ; C, DOWN
FC3A: F0 DE      142         BEQ UP      ; D, GO UP
```

```
FC3C  69 FD        143          ADC #$FD    ; ESC-E OR F CKECK
FC3E  90 5C        144          BCC CLREOL  ; E. CLEAR TO END OF LINE
FC40  D0 E9        145          BNE RTS4    ; ELSE NOT F.RETURN
FC42  A4 24        146 CLREOP   LDY CH      ; ESC F IS CLR TO END OF PAGE
FC44  A5 25        147          LDA CV        (CV)放到(A)內。
FC46  48           148 CLEOP1   PHA           (A)放到堆疊器內。
FC47  20 24 FC     149          JSR VTABZ叫用VTABE以算出BASL.H對應於(A)與WNDLFT
FC4A  20 9E FC     150          JSR CLEOLZ叫用CLEOLZ以清除CV列，CH列WNDWDTH
FC4D  A0 00        151          LDY #$00  ΦΦ放入(Y)。                    爲空白。
FC4F  68           152          PLA   (A)從堆疊器中放回。  由於此時carry一定爲1；因爲從199
FC50  69 00        153          ADC #$00      式返回時carry一定爲1，故(A)加1放到(A)內。
FC52  C5 23        154          CMP WNDBTM  比較是否要被清除的列已經到WNDBTM。
FC54  90 F0        155          BCC CLEOP1  不是的話，則至CLEOP1繼續清除畫面。
FC56  B0 CA        156          BCS VTAB    是的話，則算出游標目前對應記憶體位置。
FC58  A5 22        157 HOME     LDA WNDTOP  (A)內放捲動畫面的頂線號碼。
FC5A  85 25        158          STA CV        (A)移到CV內。
FC5C  A0 00        159          LDY #$00      (Y)內放ΦΦ。
FC5E  84 24        160          STY CH 將(Y)放到CH內。此時游標的位置移至捲動畫面的左上方。
FC60  F0 E4        161          BEQ CLEOP1 跳至CLEOP1。
FC62               162          PAGE      .
FC62  A9 00        163 CR       LDA #$00    零放到(A)內。
FC64  85 24        164          STA CH      (A)放到CH內。
FC66  E6 25        165 LF       INC CV      CV加1。
FC68  A5 25        166          LDA CV      (CV)放到(A)內。
FC6A  C5 23        167          CMP WNDBTM (A)與WNDBTM比較。檢查游標是否已經超過了底部。
FC6C  90 B6        168          BCC VTABZ   如果(A)小於WNDBTM則跳至VTABZ。
FC6E  C6 25        169          DEC CV    此時已到底部了，故要做捲動動作。CV減1，以指向
FC70  A5 22        170 SCROLL   LDA WNDTOP WNDTOP內容放到(A)內。    WNDBTM的上一行。
FC72  48           171          PHA   將(A)放到堆疊器上。
FC73  20 24 FC     172          JSR VTABZ 叫用VTABZ，以便算出頂線左邊對應的記憶體位置。
FC76  A5 28        173 SCRL1    LDA BASL  將BASL.H移至BAS2L，2H內，以便BAS2L，2H
FC78  85 2A        174          STA BAS2L 做爲捲動畫面時的目的地列指標。
FC7A  A5 29        175          LDA BASH    (WNDWDTH) 放至(Y)內。
FC7C  85 2B        176          STA BAS2H   (Y)減1放至Y內。
FC7E  A4 21        177          LDY WNDWDTH 從堆疊器放回A值。
FC80  88           178          DEY         (A)+1+CAKRY 放到(A)。此時(A)爲列源的號碼。
FC81  68           179          PLA         由於第一次進入時carry爲1，故第一個捲動移動距離
FC82  69 01        180          ADC #$01    爲兩列，以後carry皆爲0，故移動距離爲一列。
FC84  C5 23        181          CMP WNDBTM  判斷(A)是過大於或等於WNDWDTH。
FC86  B0 0D        182          BCS SCRL3   如果是的話，則跳至SCRL3，以清除WNDBTM那一列。
FC88  48           183          PHA         將(A)的放在堆疊器上。
FC89  20 24 FC     184          JSR VTABZ
FC8C  B1 28        185 SCRL2    LDA (BASL),Y 叫用VTABZ，以算出列源左邊對應的記憶體位置。
FC8E  91 2A        186          STA (BAS2L),Y   將（BASL.H）+Y處的值放到（BAS2L.2H）+Y處
FC90  88           187          DEY           記憶體內。故從一列移至另一列是從右邊開始移。
FC91  10 F9        188          BPL SCRL2  跳至SCRL2，表示尚未移完。
FC93  30 E1        189          BMI SCRL1  一列已移完，故跳至SCRL1。
FC95  A0 00        190 SCRL3    LDY #$00  要將WNDBTM這列整列清除爲空白。故ΦΦ放入(Y)。
FC97  20 9E FC     191          JSR CLEOLZ  叫用CLEOLZ，以清除掉WNDBTM這一列。
FC9A  B0 B6        192          BCS VTAB  由於CLEOLZ返回時，carry爲1，故P跳至VTAB，以算出
FC9C  A4 24        193 CLREOL   LDY CH
FC9E  A9 A0        194 CLEOLZ   LDA #$A0        (CH)放到(Y)內。    （BASL.H）此時對應的位置。
FCA0  91 28        195 CLEOL2   STA (BASL),Y    將（BASL.H）+Y處的憶體填入空白。
FCA2  C8           196          INY (Y)加1。    將空白的ASCII碼放到(A)內。
FCA3  C4 21        197          CPY WNDWDTH 判斷Y是否大於或等於捲動畫面的右邊邊緣。
FCA5  90 F9        198          BCC CLEOL2 如果是小於WNDWDTH，則跳至195行，以便繼續填空白。
FCA7  60           199          RTS  返回。
FCA8  38           200 WAIT     SEC  此段程式要參看第四章等常規的公式。
FCA9  48           201 WAIT2    PHA  在叫用（JSR）WAIT，到執行WAIT的RTS時，所需的時間爲
FCAA  E9 01        202 WAIT3    SBC #$01  (2.5*(A(2)+13.5*A+MC)*MS。
FCAC  D0 FC        203          BNE WAIT3  A=ACC的內容。
FCAE  68           204          PLA   MC=13機械週期。
FCAF  E9 01        205          SBC #$01  MS=1.Φ23微秒。
FCB1  D0 F6        206          BNE WAIT2
FCB3  60           207          RTS
FCB4  E6 42        208 NXTA4    INC A4L  增加（A4L）
FCB6  D0 02        209          BNE NXTA1  如爲零則增加（A4H）
FCB8  E6 43        210          INC A4H
FCBA  A5 3C        211 NXTA1    LDA A1L 比較（A1L.H）和（A2L.H）。如果（A1L.H）＞（A2L.H）
FCBC  C5 3E        212          CMP A2L  則設定carry。
FCBE  A5 3D        213          LDA A1H
FCC0  E5 3F        214          SBC A2H
FCC2  E6 3C        215          INC A1L
```

```
FCC4  D0 02        216           BNE RTS4B   遞加（A1L,H），故當（A1L,H）>（A2L,H）
FCC6  E6 3D        217           INC A1H       則（A1L,H）便加2。
FCC8  60           218 RTS4B     RTS
FCC9               219           PAGE
FCC9  A0 4B        220 HEADR     LDY #$4B    寫一10秒的同步單一頻率至磁帶上。
FCCB  20 DB FC     221           JSR ZERDLY
FCCE  D0 F9        222           BNE HEADR
FCD0  69 FE        223           ADC #$FE
FCD2  B0 F5        224           BCS HEADR
FCD4  A0 21        225           LDY #$21
FCD6  20 DB FC     226 WRBIT     JSR ZERDLY   寫一個位元到磁帶上。叫用ZERDLY，以做一延遲。
FCD9  C8           227           INY  (Y)變成$2D。
FCDA  C8           228           INY  (Y)變成$2E。再做一固定的迴圈延遲。
FCDB  88           229 ZERDLY    DEY          做一延遲，然後根據carry判斷寫Φ或1
FCDC  D0 FD        230           BNE ZERDLY   如carry＝1，則再做延遲，再觸發TAPEOUT。
FCDE  90 05        231           BCC WRTAPE
FCE0  A0 32        232           LDY #$32
FCE2  88           233 ONEDLY    DEY                                  讀入鍵盤輸入後中斷，
FCE3  D0 FD        234           BNE ONEDLY                           並與(A)做AND動作，
FCE5  AC 20 C0     235 WRTAPE    LDY TAPEOUT  將磁帶資料，做一變化動作。  此時(A)爲游標所指的
FCE8  A0 2C        236           LDY #$2C     (Y)放$2C。              字尚未成爲閃爍字元的
FCEA  CA           237           DEX          (X)－1。                 碼。故第7個位元一定
FCEB  60           238           RTS          返回。                   爲1。
FCEC  A2 08        239 RDBYTE    LDX #$08   一個位元組共有8個位元。
FCEE  48           240 RDBYT2    PHA    將(A)貯放在堆疊器中。
FCEF  20 FA FC     241           JSR RD2BIT  叫用RD2BIT，以讀取一個位元。
FCF2  68           242           PLA    放回(A)。
FCF3  2A           243           ROL A  右旋經過carry，將255式的結果carry，旋轉入(A)內。
FCF4  A0 3A        244           LDY #$3A  Φ重定(Y)值。
FCF6  CA           245           DEX    是否讀滿一個位元組了。
FCF7  D0 F5        246           BNE RDBYT2  未讀滿，則跳至RDBYT2。
FCF9  60           247           RTS    返回。
FCFA  20 FD FC     248 RD2BIT    JSR RDBIT  叫用RDBIT兩次，每次都必須TAPEIN有變化（結果
FCFD  88           249 RDBIT     DEY    利用Y來做爲找TAPEIN變化的時間，以決定出資料爲Φ或1
FCFE  AD 60 C0     250           LDA TAPEIN  取出TAPEIN。
FD01  45 2F        251           EOR LASTIN  看是否有變化。
FD03  10 F8        252           BPL RDBIT   如無，則返回RDBIT。
FD05  45 2F        253           EOR LASTIN  恢復新的TAPEIN值。
FD07  85 2F        254           STA LASTIN  放到LASTIN內。
FD09  C0 80        255           CPY #$80  比較(Y)與$80，以做爲判斷尋找的時間，以決定出Φ或1
FD0B  60           256           RTS 返回。
FD0C  A4 24        257 RDKEY     LDY CH  將(CH)的值放到Y暫存器內，做(Y)爲游標的水平位置
FD0E  B1 28        258           LDA (BASL),Y  取出游標位置的記憶體內容。
FD10  48           259           PHA   將(A)放到堆疊器上，以便在270式時，將字元放回去。
FD11  29 3F        260           AND #$3F  與下式將(A)變成爲對應的閃爍字元。參看ASCII表。
FD13  09 40        261           ORA #$40      將此閃爍字元放回到游標處的記憶體位置。
FD15  91 28        262           STA (BASL),Y  將(A)從堆疊器上取回。
FD17  68           263           PLA           碼KSWL,H的內容，做一間接跳的動作。
FD18  6C 38 00     264           JMP (KSWL)    （KSWL,H）一般是指向KEYIN。
FD1B  E6 4E        265 KEYIN     INC RNDL    此三行的目的爲KEYIN每執行一次，（RNDL,H）
FD1D  D0 02        266           BNE KEYIN2    加1，以便（RNDL,H）成爲亂數的產生欄。
FD1F  E6 4F        267           INC RNDH
FD21  2C 00 C0     268 KEYIN2    BIT KBD     ; READ KEYBOARD
FD24  10 F5        269           BPL KEYIN 如果爲正數，則返回到KEYIN即表示無鍵盤的Scrobe信號。
FD26  91 28        270           STA (BASL),Y  將游標所指的閃爍字元，恢復爲沒閃爍的字元放回。
FD28  AD 00 C0     271           LDA KBD  將鍵盤輸入緩衝器的值，放到(A)內。
FD2B  2C 10 C0     272           BIT KBDSTRB  清除掉鍵盤的觸發信號。
FD2E  60           273           RTS  返回到RDCHAR。
FD2F  20 0C FD     274 ESC       JSR RDKEY  叫用RDKEY，以便讀取下一個鍵入的字元。
FD32  20 A5 FB     275           JSR ESCNEW  叫用ESCNEW，以便處理下一個鍵入的新字元。
FD35  20 0C FD     276 RDCHAR    JSR RDKEY  叫用RDKEY，以便得到下一個字元，放到(A)內。
FD38  C9 9B        277           CMP #$9B  比較是否爲ESC鍵被按下去？
FD3A  F0 F3        278           BEQ ESC  是"ESC"碼，則跳至ESC常規，而做ESC的功能。
FD3C  60           279           RTS 返回。
FD3D               280           PAGE
FD3D  A5 32        281 NOTCR     LDA INVFLG  與下式將INVFLG貯放在堆疊器上。
FD3F  48           282           PHA
FD40  A9 FF        283           LDA #$FF  與下式將$FF貯放到INVFLG。
FD42  85 32        284           STA INVFLG
FD44  BD 00 02     285           LDA IN,X  將放在（Φ2ΦΦ+X）內的字元放到(A)內。
FD47  20 ED FD     286           JSR COUT  叫用COUT以出此字元。
FD4A  68           287           PLA  與下式恢復原來INVFLG的值。
FD4B  85 32        288           STA INVFLG
```

```
FD4D: BD 00 02    289           LDA IN,X 再判定（IN＋X）的字元是否爲向左移或列取消（控制-X）。
FD50: C9 88       290           CMP #$88 如果是的話則跳至BCKSPC或CANCEL。
FD52: F0 1D       291           BEQ BCKSPC          如果不跳至BCKSPC或CANCEL，則判斷X
FD54: C9 98       292           CMP #$9B            （即輸入的字元數目）是否大於247。
FD56: F0 0A       293           BEQ CANCEL
FD58: E0 F8       294           CPX #$F8          (X)小於247，則跳至NOTCR1。
FD5A: 90 03       295           BCC NOTCR1        (X)大於或等於247則叫用BELL，以發出警聲。
FD5C: 20 3A FF    296           JSR BELL          (X)＋1以指向下一個輸入字元要貯放的位置。
FD5F: E8          297 NOTCR1    INX               如果不爲零，表示無滿溢現像則跳至NXTCHAR。
FD60: D0 13       298           BNE NXTCHAR         (A)放到“/”的ASCII碼。
FD62: A9 DC       299 CANCEL    LDA #$DC          叫用COUT以印出“/”，然後再進入GETLNZ，
FD64: 20 ED FD    300           JSR COUT          以便下一行的輸入。
FD67: 20 8E FD    301 GETLNZ    JSR CROUT 印出carriage return到螢幕上。
FD6A: A5 33       302 GETLN     LDA PROMPT 從（33）處取出提醒字元“*”的ASCII碼，放到A-暫存
FD6C: 20 ED FD    303           JSR COUT 將(A)的字元印到螢幕上。                器內。
FD6F: A2 01       304           LDX #$01 (X)被設定的初值爲φ1，經307式，則(X)的初值爲0。
FD71: BA          305 BCKSPC    TXA (X)放到(Y)內。
FD72: F0 F3       306           BEQ GETLNZ 判斷(X)是否爲零。如果爲φ則跳至GETLNZ以便重新讀
FD74: CA          307           DEX (X)減1。                                    一行資料。
FD75: 20 35 FD    308 NXTCHAR   JSR RDCHAR 叫用RDCHAR，以便得到一個字元，放在(A)裏面。
FD78: C9 95       309           CMP #$95 比較(A)內是否爲控制-U，即右箭頭。
FD7A: D0 02       310           BNE CAPTST 不是的話，則跳至CAPTST。
FD7C: B1 28       311           LDA (BASL),Y 是的話，則從（BASL,H）＋Y處拿出字元，故(A)爲
FD7E: C9 E0       312 CAPTST    CMP #$E0 判斷此字元是否爲大寫字元。            游標前一個的字元碼。
FD80: 90 02       313           BCC ADDINP 是的話，則跳至ADDINP。
FD82: 29 DF       314           AND #$DF      SHIFT TO UPPER CASE不是的話，則轉換成大寫。
FD84: 9D 00 02    315 ADDINP    STA IN,X 將從鍵盤讀入的字元放在IN＋X處。
FD87: C9 8D       316           CMP #$8D 判斷此字元是否爲carriage return（$8D）。
FD89: D0 B2       317           BNE NOTCR 不是的話，則跳至NOTCH。
FD8B: 20 9C FC    318           JSR CLREOL 是的話，則叫用CLREOL，以消除掉游標至該列的最右方。
FD8E: A9 8D       319 CROUT     LDA #$8D 經由COUT印出carriage return。
FD90: D0 5B       320           BNE COUT
FD92: A4 3D       321 PRA1      LDY A1H （A1H）放到(Y)內。
FD94: A6 3C       322           LDX A1L （A1L）放到(X)內。
FD96: 20 8E FD    323 PRYX2     JSR CROUT 印出carriage return。
FD99: 20 40 F9    324           JSR PRNTYX 印出Y，X的十六進位值。
FD9C: A0 00       325           LDY #$00 φφ放入Y中，此值在340式用到。
FD9E: A9 AD       326           LDA #$AD “－”的碼放入(A)內。
FDA0: 4C ED FD    327           JMP COUT 跳至COUT，以印出“－”。
FDA3:             328           PAGE
FDA3: A5 3C       329 XAM8      LDA A1L             將A1L的右半部變成爲7，然後再存入A2L，
FDA5: 09 07       330           ORA #$07            故（A1L.H）的內容爲mmmm。而（A2L.H）
FDA7: 85 3E       331           STA A2L             的內容則爲mmm 7。
FDA9: A5 3D       332           LDA A1H
FDAB: 85 3F       333           STA A2H           檢查A1L的後三個位元是否皆爲零了，如果是，則叫用
FDAD: A5 3C       334 MOD8CHK   LDA A1L           φPRA1以印出（A1L.H），“－”及CR。
FDAF: 29 07       335           AND #$07            印出一個空白。
FDB1: D0 03       336           BNE DATAOUT
FDB3: 20 92 FD    337 XAM       JSR PRA1            將（A1L.H）所指的地方的資料拿出，放到(A)內。
FDB6: A9 A0       338 DATAOUT   LDA #$A0            此時Y在325式中已爲零。叫用PRBYTE，以印出
FDB8: 20 ED FD    339           JSR COUT            (A)內的兩個十六進位字元。
FDBB: B1 3C       340           LDA (A1L),Y
FDBD: 20 DA FD    341           JSR PRBYTE          叫用NXTA1，如果（A2L.H）＜（A1L.H），則
FDC0: 20 BA FC    342           JSR NXTA1           設定carry旗號。
FDC3: 90 E8       343           BCC MOD8CHK 如（A1L.H）不大於（A2L.H），則（A1L.H）加1，
FDC5: 60          344 RTS4C     RTS 返回。                                      而返回MDD8CHK。
FDC6: 4A          345 XAMPM     LSR A (A)的LSB右移到carry內。
FDC7: 90 EA       346           BCC XAM 如果carry爲φ，表示mode＝“.”，故跳至XAM，以做
FDC9: 4A          347           LSR A               顯示記憶體的動作。
FDCA: 4A          348           LSR A 剩下“＋”與“－”的可能，故右移兩次，以判斷爲“＋”或
FDCB: A5 3E       349           LDA A2L 將加數或減數放到（A2L）內。                “－”。
FDCD: 90 02       350           BCC ADD 如爲“＋”則跳至ADD。
FDCF: 49 FF       351           EOR #$FF 做1'S互補。
FDD1: 65 3C       352 ADD       ADC A1L（A1L）＋（A2L）＋carry或（A1L）＋（A2L 2的補數）。
FDD3: 48          353           PHA                 將結果放於堆疊器中。
FDD4: A9 BD       354           LDA #$BD            “＝”的碼放在(A)內。
FDD6: 20 ED FD    355           JSR COUT 叫用COUT，以印出＝。
FDD9: 68          356           PLA                 將結果放在(A)內，再由下面印出。
FDDA: 48          357 PRBYTE    PHA 將一個位元組，以二個十六進位字元印出。
FDDB: 4A          358           LSR A               右轉四次，以將左半部移至右半部。
FDDC: 4A          359           LSR A
FDDD: 4A          360           LSR A
FDDE: 4A          361           LSR A
```

```
FDDF  20 E5 FD    362          JSR PRHEXZ   叫用PRHEXZ，以轉換成16進位，然後再印出。
FDE2  68          363          PLA    從堆疊器中取出(A)值。
FDE3  29 0F       364 PRHEX    AND #$0F   取出右半部。
FDE5  09 B0       365 PRHEXZ   ORA #$B0   轉換成ASCII碼。
FDE7  C9 BA       366          CMP #$BA   如比$BA小，表示0到9，則可直接印出。如大於或等於
FDE9  90 02       367          BCC COUT   $BA，表示要印出A到F 則加上06再加上carry，故總加上 φ7，
FDEB  69 06       368          ADC #$06   以轉換成ASCII碼。
FDED  6C 36 00    369 COUT     JMP (CSWL)  經由（CSWL）的內容，跳至一對應輸出裝置常規。
FDF0  C9 A0       370 COUT1    CMP #$A0   與空白比較。                一般是指向COUT1。
FDF2  90 02       371          BCC COUTZ 小於空白碼，則跳至COUTZ。
FDF4  25 32       372          AND INVFLG 根據INVFLG，將要輸出的字元轉換成適當的碼。
FDF6  84 35       373 COUTZ    STY YSAV1 將(Y)貯放於保留區域YSAV1。
FDF8  48          374          PHA   將(A)放入堆疊器。
FDF9  20 78 FB    375          JSR VIDWAIT ; GO CHECK FOR PAUSE 叫用VIDWAIT，測試操
FDFC  68          376          PLA    從堆疊器上取回(A)值。                做者是否要求暫停。
FDFD  A4 35       377          LDY YSAV1  恢復Y值。
FDFF  60          378          RTS    返回。
FE00              379          PAGE
FE00  C6 34       380 BL1      DEC YSAV    如果*後面打字CR，則在571式時Y＝1，故在此動作後
FE02  F0 9F       381          BEQ XAM8 跳至XAM8，以顯示mmmm至mmm8的資料。    Y＝0。
FE04  CA          382 BLANK    DEX    當從TO$UB過來時X－1；(A)＝（MODE）。
FE05  D0 16       383          BNE SETMDZ  如果(X)不等於1，則跳至SETMDZ。
FE07  C9 BA       384          CMP #$BA   (A)是否為：號。
FE09  D0 BB       385          BNE XAMPM  不是的話，則跳至XAMPM。
FE0B  85 31       386 STOR     STA MODE  此段為（mode）＝":"。將（A2L）放到（A3L）+Y
FE0D  A5 3E       387          LDA A2L    所指的位置。
FE0F  91 40       388          STA (A3L),Y
FE11  E6 40       389          INC A3L    增加（A3L.H）。
FE13  D0 02       390          BNE RTS5
FE15  E6 41       391          INC A3H
FE17  60          392 RTS5     RTS    返回。
FE18  A4 34       393 SETMODE  LDY YSAV    將(Y)從YSAV取回。
FE1A  B9 FF 01    394          LDA IN-1,Y   從輸入資料中取回命令，由於Y指向命令的下一個字，
FE1D  85 31       395 SETMDZ   STA MODE    故IN－1＋Y。取回的資料放到MODE中。
FE1F  60          396          RTS    返回。
FE20  A2 01       397 LT       LDX #$01  將（A2L.H）的內容移入（A4L.H）及（A5L.H）中。
FE22  B5 3E       398 LT2      LDA A2L,X
FE24  95 42       399          STA A4L,X
FE26  95 44       400          STA A5L,X
FE28  CA          401          DEX
FE29  10 F7       402          BPL LT2      此段是將（A1L.H）到（A2L.H）的資料移至
FE2B  60          403          RTS          （A4L.H）處開始。
FE2C  B1 3C       404 MOVE     LDA (A1L),Y
FE2E  91 42       405          STA (A4L),Y    叫用NXTA4，增加（A4L.H）和比較（A1L.H）
FE30  20 B4 FC    406          JSR NXTA4      和（A2L.H）及（A1L.H）加1。
FE33  90 F7       407          BCC MOVE 尚未移到（A2L.H），故跳回MOVE再移。
FE35  60          408          RTS   返回。
FE36  B1 3C       409 VFY      LDA (A1L),Y   比較（A1L.H）與（A4L.H）所指的內容如相等，
FE38  D1 42       410          CMP (A4L),Y   則跳至VFYOK。
FE3A  F0 1C       411          BEQ VFYOK
FE3C  20 92 FD    412          JSR PRA1 印出（A1L.H）和"－"。
FE3F  B1 3C       413          LDA (A1L),Y 取出（A1L.H）所指的內容，然後叫用PRBYTE印出。
FE41  20 DA FD    414          JSR PRBYTE
FE44  A9 A0       415          LDA #$A0  經COUT印出一個空白。
FE46  20 ED FD    416          JSR COUT
FE49  A9 A8       417          LDA #$A8  印出"("。
FE4B  20 ED FD    418          JSR COUT
FE4E  B1 42       419          LDA (A4L),Y 印出（A4L.H）所指的內容。
FE50  20 DA FD    420          JSR PRBYTE
FE53  A9 A9       421          LDA #$A9 印出")"。
FE55  20 ED FD    422          JSR COUT       叫用NXTA4，以增加（A4L.H）及（A1L.H），
FE58  20 B4 FC    423 VFYOK    JSR NXTA4      且比較（A1L.H）和（A2L.H）。
FE5B  90 D9       424          BCC VFY  （A2L.H）小於（A1L.H）則跳回VFY，以繼續驗證比較。
FE5D  60          425          RTS  返回。
FE5E  20 75 FE    426 LIST     JSR A1PC.L 命令的進入點。叫用A1PC，以便（A1L.H）放到（PCL.H）。
FE61  A9 14       427          LDA #$14  (A)放$14，以便每次反組合20個指令。
FE63  48          428 LIST2    PHA  將(A)放入堆疊器中，以免下面破壞掉。
FE64  20 D0 F8    429          JSR INSTDSP 叫用INSTDSP，以反組合指令，並且印出。
FE67  20 53 F9    430          JSR PCADJ 叫用PCADJ，以調整下個要被反組合指令的位址。
FE6A  85 3A       431          STA PCL  將調整過的位址，放到PCL.H中。
FE6C  84 3B       432          STY PCH
FE6E  68          433          PLA  取回(A)。
FE6F  38          434          SEC  設定carry。故在下式SBC變造成(A)＝(A)－1而已。
```

```
FE70: E9 01       435          SBC #$01
FE72: D0 EF       436          BNE LIST2  尚未反組合到第20個指令，則跳回到LIST2。
FE74: 60          437          RTS  返回。
FE75:             438          PAGE  (X)放到(A)內。          將（A1L,H）移到（PCL,H），
FE75: 8A          439 A1PC     TXA    如爲零，則返回。          先移PCH，再移PCL。故(X)應
FE76: F0 07       440          BEQ A1PCRTS                    爲1進入。
FE78: B5 3C       441 A1PCLP   LDA A1L,X  A1PCLP。
FE7A: 95 3A       442          STA PCL,X
FE7C: CA          443          DEX
FE7D: 10 F9       444          BPL A1PCLP
FE7F: 60          445 A1PCRTS  RTS  返回。
FE80: A0 3F       446 SETINV   LDY #$3F  (Y)放入$3F，以設定螢幕爲白底黑字。
FE82: D0 02       447          BNE SETIFLG  跳至SETIFLG。
FE84: A0 FF       448 SETNORM  LDY #$FF  (Y)放入$FF，以設定螢幕爲黑底白字。
FE86: 84 32       449 SETIFLG  STY INVFLG  將(Y)放入INVFLG，以決定螢幕的底色。
FE88: 60          450          RTS  返回。
FE89: A9 00       451 SETKBD   LDA #$00  設定埠0（鍵盤）爲輸入。
FE8B: 85 3E       452 INPORT   STA A2L  (A)放到A2L。
FE8D: A2 38       453 INPRT    LDX #KSWL  #KSWL放到(X)內。#KSWL在程式前面定義。
FE8F: A0 1B       454          LDY #KEYIN  #KEYIN放到(Y)內。#KEYIN則在265式定義。
FE91: D0 08       455          BNE IOPRT  跳至IOPRT。
FE93: A9 00       456 SETVID   LDA #$00  設定埠0（螢幕）爲輸出。
FE95: 85 3E       457 OUTPORT  STA A2L  (A)放到A2L。
FE97: A2 36       458 OUTPRT   LDX #CSWL  #CSWL放到(X)內。#CSWL在程式前面定義。
FE99: A0 F0       459          LDY #COUT1  #COUT1放到(Y)內。#COUT1則在370式定義。
FE9B: A5 3E       460 IOPRT    LDA A2L  取出A2L，放到(A)內。如不爲0，則（A2L）的值是從585式
FE9D: 29 0F       461          AND #$0F  取出(A)內的右半部。                    放入的。
FE9F: F0 06       462          BEQ IOPRT1  如爲0，則跳至IOPRT1。
FEA1: 09 C0       463          ORA #IOADR/256 取出對應於槽號0的週邊控制卡PROM的位置。
FEA3: A0 00       464          LDY #$00  由$C100至$C700。
FEA5: F0 02       465          BEQ IOPRT2  跳至IOPRT2。
FEA7: A9 FD       466 IOPRT1   LDA #COUT1/256  (A)放入$FD。
FEA9:             467 IOPRT2   EQU *
FEA9: 94 00       468          STY LOC0,X ; $94,$00  將(Y)放入LOC0+X內。
FEAB: 95 01       469          STA LOC1,X ; $95,$01  將(A)放入LOC1+X內。
FEAD: 60          470          RTS  返回。
FEAE: EA          471          NOP
FEAF: EA          472          NOP
FEB0: 4C 00 E0    473 XBASIC   JMP BASIC  進入BASIC語言。
FEB3: 4C 03 E0    474 BASCONT  JMP BASIC2  再進入BASIC語言內。
FEB6: 20 75 FE    475 GO       JSR A1PC  叫用A1PC，以設定（PCL,H）爲命令G的起始位置。
FEB9: 20 3F FF    476          JSR RESTORE  叫用RESTORE，以恢復S暫存器以外的所有暫存器。
FEBC: 6C 3A 00    477          JMP (PCL)  經由（PCL,H）間接跳而開始執行。
FEBF: 4C D7 FA    478 REGZ     JMP REGDSP 跳至REGDSP，以印出CR，及顯示暫存器內容。
FEC2: 60          479 TRACE    RTS        T命令在自動開始的監督程式被視爲錯誤的命令，故及S
FEC3:             480 * TRACE  IS GONE  在573式CHRSRCH中，會跳回MON。
FEC3: EA          481          NOP
FEC4: 60          482 STEPZ    RTS              STEP IS GONE
FEC5: EA          483          NOP
FEC6: EA          484          NOP
FEC7: EA          485          NOP
FEC8: EA          486          NOP
FEC9: EA          487          NOP
FECA: 4C F8 03    488 USR      JMP USRADR 跳至USRADR，以執行使用者定義的控制-Y動作。
FECD:             489          PAGE
FECD: A9 40       490 WRITE    LDA #$40 寫一個10秒鐘的同步單頻，以做爲前面標誌。
FECF: 20 C9 FC    491          JSR HEADR  此常規中，(Y)值決定出信號的頻率及同步之用。
FED2: A0 27       492          LDY #$27
FED4: A2 00       493 WR1      LDX #$00  X放00。
FED6: 41 3C       494          EOR (A1L,X) 做EOR以得出checksum，然後貯放在堆疊器中。
FED8: 48          495          PHA
FED9: A1 3C       496          LDA (A1L,X) 取出要寫的位元組。
FEDB: 20 ED FE    497          JSR WRBYTE  叫用WRBYTE寫出。
FEDE: 20 BA FC    498          JSR NXTA1  叫用NXTA1，檢查是否寫到（A2L,H）了。
FEE1: A0 1D       499          LDY #$1D
FEE3: 68          500          PLA  取出checksum。
FEE4: 90 EE       501          BCC WR1  如尚未寫到（A2L,H），則跳回WR1，繼續寫。
FEE6: A0 22       502          LDY #$22
FEE8: 20 ED FE    503          JSR WRBYTE  將checksum寫出。
FEEB: F0 4D       504          BEQ BELL  跳至BELL，以響一鈴聲和出現凹顯字元。
FEED: A2 10       505 WRBYTE   LDX #$10  寫一個位元組，共十個位元至磁帶上。
FEEF: 0A          506 WRBYT2   ASL A  將MSB移入carry，以決定要是寫0或1。
FEF0: 20 D6 FC    507          JSR WRBIT  叫用WRBIT，以寫一個位元到磁帶上。
```

```
FEF3: D0 FA       508          BNE WRBYT2 在237式中DEX，由此判斷是否寫完了10個位元。
FEF5: 60          509          RTS 返回。          carriage return 命令的進入點。叫用BL1，
FEF6: 20 00 FE    510 CRMON    JSR BL1             以模擬一個空白命令。
FEF9: 68          511          PLA 從堆疊器取出二個位元組，此二位元組即577式的返回位址。
FEFA: 68          512          PLA
FEFB: D0 6C       513          BNE MONZ 跳至MONZ。
FEFD: 20 FA FC    514 READ     JSR RD2BIT 叫用RD2BIT，以讀到有資料爲止。
FF00: A9 16       515          LDA #$16 做3.5秒的延遲。
FF02: 20 C9 FC    516          JSR HEADR
FF05: 85 2E       517          STA CHKSUM (A)放入checksum內。
FF07: 20 FA FC    518          JSR RD2BIT 叫用RD2BIT，以讀取兩個位元。
FF0A: A0 24       519 RD2      LDY #$24 (Y)賦放$24。
FF0C: 20 FD FC    520          JSR RDBIT 叫用RDBIT，以讀取一個位元。
FF0F: B0 F9       521          BCS RD2 如(Y)時大於或等於$80，則跳回RD2，此位元爲1。
FF11: 20 FD FC    522          JSR RDBIT 叫用RDBIT，以讀取一個位元。
FF14: A0 3B       523          LDY #$3B (Y)賦放$3B。
FF16: 20 EC FC    524 RD3      JSR RDBYTE 叫用RDBYTE，以讀取一個位元組。
FF19: 81 3C       525          STA (A1L,X) 暫放在 (A1L,H)。
FF1B: 45 2E       526          EOR CHKSUM 決定出新的checksum。
FF1D: 85 2E       527          STA CHKSUM 將新的checksum放回。
FF1F: 20 BA FC    528          JSR NXTA1 叫用NXTA1，判斷是否讀完了。
FF22: A0 35       529          LDY #$35 (Y)放$35。
FF24: 90 F0       530          BCC RD3 如尚未讀完，則跳回RD3。
FF26: 20 EC FC    531          JSR RDBYTE 再讀一個位元組。
FF29: C5 2E       532          CMP CHKSUM 比較checksum 是否符合。
FF2B: F0 0D       533          BEQ BELL 如符合，則響蜂叫，否則印出ERR。
FF2D: A9 C5       534 PRERR    LDA #$C5 將"E"的ASCII碼放到(A)內。
FF2F: 20 ED FD    535          JSR COUT 叫用COUT印出。
FF32: A9 D2       536          LDA #$D2 將"R"的ASCII碼，放到(Y)內。
FF34: 20 ED FD    537          JSR COUT 叫用COUT印出。
FF37: 20 ED FD    538          JSR COUT 叫用COUT印出，故印出ERR三個字。
FF3A: A9 87       539 BELL     LDA #$87 將控制-G，即鈴響碼經由COUT印出，故會響一聲。
FF3C: 4C ED FD    540          JMP COUT
FF3F:             541          PAGE 除了S 暫存器以外的暫存器在監督程式命令G之前，從此處恢復。
FF3F: A5 48       542 RESTORE  LDA STATUS 將狀態放在(A)內，再移到堆疊器內。
FF41: 48          543          PHA
FF42: A5 45       544          LDA A5H 恢復(A)。
FF44: A6 46       545 RESTR1   LDX XREG 恢復(X)。
FF46: A4 47       546          LDY YREG 恢復(Y)。
FF48: 28          547          PLP 從堆疊器上恢復狀態暫存器。
FF49: 60          548          RTS 返回。
FF4A: 85 45       549 SAVE     STA A5H 保留(A)。
FF4C: 86 46       550 SAV1     STX XREG 保留(X)。
FF4E: 84 47       551          STY YREG 保留(Y)。
FF50: 08          552          PHP 保留(P)。
FF51: 68          553          PLA
FF52: 85 48       554          STA STATUS
FF54: BA          555          TSX 保留堆疊器指標。
FF55: 86 49       556          STX SPNT
FF57: D8          557          CLD 清除十進位模式。
FF58: 60          558          RTS 返回。
FF59: 20 84 FE    559 OLDRST   JSR SETNORM         爲了與舊型監督程式互用，此程
FF5C: 20 2F FB    560          JSR INIT            式仍然存在。此與557式同。
FF5F: 20 93 FE    561          JSR SETVID
FF62: 20 89 FE    562          JSR SETKBD
FF65:             563          PAGE
FF65: D8          564 MON      CLD 清除65φ2的十進位模式。
FF66: 20 3A FF    565          JSR BELL 響鈴響。
FF69: A9 AA       566 MONZ     LDA #$AA 將"*"的ASCII碼放入PROMPT內。
FF6B: 85 33       567          STA PROMPT
FF6D: 20 67 FD    568          JSR GETLNZ 叫用GETLNZ以讀整個命令線。
FF70: 20 C7 FF    569          JSR ZMODE 在掃描命令線之前，清除MODE位元組。
FF73: 20 A7 FF    570 NXTITM   JSR GETNUM 叫用GETNUM，以得到一16進位數目字於(A2L,H)中。
FF76: 84 34       571          STY YSAV (A)含命令，(Y)爲命令的下一個字，即目前命令列上，下一個要
FF78: A0 17       572          LDY #$17 被分析的字元。開始分析命令，共有23個。
FF7A: 88          573 CHRSRCH  DEY 從64式開始比較命令，如果沒有，則往上逐一比較，當比較超過23次時，
FF7B: 30 E8       574          BMI MON 則跳回MON。
FF7D: D9 CC FF    575          CMP CHRTBL,Y
FF80: D0 F8       576          BNE CHRSRCH
FF82: 20 BE FF    577          JSR TOSUB 比較到對應的命令，則叫用TOSUB。
FF85: A4 34       578          LDY YSAV 恢復Y值。
FF87: 4C 73 FF    579          JMP NXTITM 跳至NXTITM，繼續拿下面的值。
FF8A: A2 03       580 DIG      LDX #$03 X放$φ3，故在NXTBIT中，要執行四次。
```

```
FF8C: 0A          581          ASL A   (A)先左移四次。
FF8D: 0A          582          ASL A
FF8E: 0A          583          ASL A
FF8F: 0A          584          ASL A          先將(A)內的值移入carry，再將ROL A2L ROL2H
FF90: 0A          585 NXTBIT   ASL A          移入（A2L.H）內。由於共移了四次，故一個十六進
FF91: 26 3E       586          ROL A2L        位的數字即移入（A2L.H）內。進入NXTBAS時，
FF93: 26 3F       587          ROL A2H        (X)=$FF。
FF95: CA          588          DEX
FF96: 10 F8       589          BPL NXTBIT     由於MODE爲零時，（A1L）先抄入（A1L）
FF98: A5 31       590 NXTBAS   LDA MODE        和（A3L）內，經598式後，（A2H）再抄入
FF9A: D0 06       591          BNE NXTBS2     （A2H）和（A3H）內。
FF9C:             592 *
FF9C: B5 3F       593          LDA A2H,X
FF9E:             594 *
FF9E: 95 3D       595          STA A1H,X
FFA0:             596 *
FFA0: 95 41       597          STA A3H,X
FFA2: E8          598 NXTBS2   INX
FFA3: F0 F3       599          BEQ NXTBAS
FFA5: D0 06       600          BNE NXTCHR 跳至NXTCHR 繼續判斷下一個字元。
FFA7: A2 00       601 GETNUM   LDX #$00 先將（A2L.H）內放入零。故當16進位不連4個時，自動在前
FFA9: 86 3E       602          STX A2L 面填入零。
FFAB: 86 3F       603          STX A2H 讀入鍵盤輸入區域的字母，第一次由於ZMODE時Y 爲零，故讀
FFAD: B9 00 02    604 NXTCHR   LDA IN,Y(φ2φφ) 的內容，以後則讀（0200＋Y）的字母。
FFB0: C8          605          INY
FFB1: 49 B0       606          EOR #$B0 如爲0至9的字母，則跳至DIG。
FFB3: C9 0A       607          CMP #$0A
FFB5: 90 D3       608          BCC DIG
FFB7: 69 88       609          ADC #$88 如果輸 的字母爲A至F ，則跳至DIG，否則返回。
FFB9: C9 FA       610          CMP #$FA
FFBB: B0 CD       611          BCS DIG
FFBD: 60          612          RTS
FFBE: A9 FE       613 TOSUB    LDA #00/256   對應程式位址的高位元組爲$φFE，放到堆疊器中。
FFC0: 48          614          PHA
FFC1: B9 E3 FF    615          LDA SUBTBL,Y 對應程式位址的低位元組爲從SUBTBL中取出，放到堆
FFC4: 48          616          PHA                                          疊器中。
FFC5: A5 31       617          LDA MODE  （MODE）放到(A)中。
FFC7: A0 00       618 ZMODE    LDY #$00  在命令之間，清除掉MODE位元組。
FFC9: 84 31       619          STY MODE 此RTS相當有技巧，從ZMODE進入時RTS返回到叫用者，當
FFCB: 60          620          RTS 從TOSUB進入時，RTS則取出前兩個PHA的內容，加1送到PC中。
FFCC:             621          PAGE 下面爲可能的$17命令，而SUBTBL則爲對應的程式位址。
FFCC: BC          622 CHRTBL   DFB #$BC FEB3 B AS CONT 控制-C
FFCD: B2          623          DFB #$B2 FECA USER 控制-Y
FFCE: BE          624          DFB #$BE FEBF REGZ 控制-E
FFCF: B2          625          DFB #$B2      T CMD NOW LIKE USR FEC2 TRACE 控制-Y
FFD0: EF          626          DFB #$EF FE36 VFY V
FFD1: C4          627          DFB #$C4 FE8D INPRT 控制-K
FFD2: B2          628          DFB #$B2      S CMD NOW LIKE USR FEC5 STEPZ＋1 控制-Y
FFD3: A9          629          DFB #$A9 FE97 OUTPRT 控制-P
FFD4: BB          630          DFB #$BB FEBφ XBASIC 控制-B
FFD5: A6          631          DFB #$A6 FE18 SETMODE -
FFD6: A4          632          DFB #$A4 FE18 SETMODE +
FFD7: 06          633          DFB #$06 FE2C MOVE M
FFD8: 95          634          DFB #$95 FE2φ LT <
FFD9: 07          635          DFB #$07 FE84 SETNORM N
FFDA: 02          636          DFB #$02 FE8φ SETINV I
FFDB: 05          637          DFB #$05 FE5E LIST L
FFDC: F0          638          DFB #$F0 FECD WRITE W
FFDD: 00          639          DFB #$00 FEB6 GO G
FFDE: EB          640          DFB #$EB FEFD READ R
FFDF: 93          641          DFB #$93 FE18 SETMODE :
FFE0: A7          642          DFB #$A7 FE18 SETMODE •
FFE1: C6          643          DFB #$C6 FEF6 CRMON carriage return
FFE2: 99          644          DFB #$99 FE04 BLANK 空白
FFE3: B2          645 SUBTBL   DFB #$B2 下面爲對應到上面命令的程式位址的低位元組。
FFE4: C9          646          DFB #$C9 NMI中斷的間接向量位置。即內容爲對應程式的位址。
FFE5: BE          647          DFB #$BE RESET中斷的間接向量位置。
FFE6: C1          648          DFB #$C1 IRQ中斷的間接向量位置。
FFE7: 35          649          DFB #$35
FFE8: 8C          650          DFB #$8C
FFE9: C4          651          DFB #$C4
FFEA: 96          652          DFB #$96
FFEB: AF          653          DFB #$AF
```

```
FFEC: 17          654       DFB $17
FFED: 17          655       DFB $17
FFEE: 2B          656       DFB $2B
FFEF: 1F          657       DFB $1F
FFF0: B3          658       DFB $B3
FFF1: 7F          659       DFB $7F
FFF2: 5D          660       DFB $5D
FFF3: CC          661       DFB $CC
FFF4: B5          662       DFB $B5
FFF5: FC          663       DFB $FC
FFF6: 17          664       DFB $17
FFF7: 17          665       DFB $17
FFF8: F5          666       DFB $F5
FFF9: 03          667       DFB $03
FFFA: FB 03       668       DW  NMI
FFFC: 62 FA       669       DW  RESET
FFFE: 40 FA       670       DW  IRQ

ENDASM
```

MONITOR ROM LISTING

```
1     ****************************
2     *                          *
3     *        APPLE II          *
4     *     SYSTEM MONITOR       *
5     *                          *
6     *   COPYRIGHT 1977 BY      *
7     *  APPLE COMPUTER, INC.    *
8     *                          *
9     *  ALL RIGHTS RESERVED     *
10    *                          *
11    *       S. WOZNIAK         *
12    *        A. BAUM           *
13    *                          *
14    ****************************
15              TITLE             "APPLE II SYSTEM MONITOR"
16    LOC0      EPZ   $00
17    LOC1      EPZ   $01
18    WNDLFT    EPZ   $20
19    WNDWDTH   EPZ   $21
20    WNDTOP    EPZ   $22
21    WNDBTM    EPZ   $23
22    CH        EPZ   $24
23    CV        EPZ   $25
24    GBASL     EPZ   $26
25    GBASH     EPZ   $27
26    BASL      EPZ   $28
27    BASH      EPZ   $29
28    BAS2L     EPZ   $2A
29    BAS2H     EPZ   $2B
30    H2        EPZ   $2C
31    LMNEM     EPZ   $2C
32    RTNL      EPZ   $2C
33    V2        EPZ   $2D
34    RMNEM     EPZ   $2D
35    RTNH      EPZ   $2D
36    MASK      EPZ   $2E
37    CHKSUM    EPZ   $2E
38    FORMAT    EPZ   $2E
39    LASTIN    EPZ   $2F
40    LENGTH    EPZ   $2F
41    SIGN      EPZ   $2F
42    COLOR     EPZ   $30
43    MODE      EPZ   $31
44    INVFLG    EPZ   $32
45    PROMPT    EPZ   $33
46    YSAV      EPZ   $34
47    YSAV1     EPZ   $35
48    CSWL      EPZ   $36
49    CSWH      EPZ   $37
50    KSWL      EPZ   $38
51    KSWH      EPZ   $39
52    PCL       EPZ   $3A
53    PCH       EPZ   $3B
54    XQT       EPZ   $3C
55    A1L       EPZ   $3C
56    A1H       EPZ   $3D
57    A2L       EPZ   $3E
58    A2H       EPZ   $3F
59    A3L       EPZ   $40
60    A3H       EPZ   $41
61    A4L       EPZ   $42
62    A4H       EPZ   $43
63    A5L       EPZ   $44
64    A5H       EPZ   $45
65    ACC       EPZ   $45
66    XREG      EPZ   $46
67    YREG      EPZ   $47
68    STATUS    EPZ   $48
```

```
                        69  SPNT      EPZ  $49
                        70  RNDL      EPZ  $4E
                        71  RNDH      EPZ  $4F
                        72  ACL       EPZ  $50
                        73  ACH       EPZ  $51
                        74  XTNDL     EPZ  $52
                        75  XTNDH     EPZ  $53
                        76  AUXL      EPZ  $54
                        77  AUXH      EPZ  $55
                        78  PICK      EPZ  $95
                        79  IN        EQU  $0200
                        80  USRADR    EQU  $03F8
                        81  NMI       EQU  $03FB
                        82  IRQLOC    EQU  $03FE
                        83  IOADR     EQU  $C000
                        84  KBD       EQU  $C000
                        85  KBDSTRB   EQU  $C010
                        86  TAPEOUT   EQU  $C020
                        87  SPKR      EQU  $C030
                        88  TXTCLR    EQU  $C050
                        89  TXTSET    EQU  $C051
                        90  MIXCLR    EQU  $C052
                        91  MIXSET    EQU  $C053
                        92  LOWSCR    EQU  $C054
                        93  HISCR     EQU  $C055
                        94  LORES     EQU  $C056
                        95  HIRES     EQU  $C057
                        96  TAPEIN    EQU  $C060
                        97  PADDL0    EQU  $C064
                        98  PTRIG     EQU  $C070
                        99  BASIC     EQU  $E000
                       100  BASIC2    EQU  $E003
                       101            ORG  $F800          ROM START ADDRESS
F800: 4A               102  PLOT      LSR  A              Y-COORD/2
F801: 08               103            PHP                 SAVE LSB IN CARRY
F802: 20 47 F8         104            JSR  GBASCALC       CALC BASE ADR IN GBASL,H
F805: 28               105            PLP                 RESTORE LSB FROM CARRY
F806: A9 0F            106            LDA  #$0F           MASK $0F IF EVEN
F808: 90 02            107            BCC  RTMASK
F80A: 69 E0            108            ADC  #$E0           MASK $F0 IF ODD
F80C: 85 2E            109  RTMASK    STA  MASK
F80E: B1 26            110  PLOT1     LDA  (GBASL),Y      DATA
F810: 45 30            111            EOR  COLOR           XOR  COLOR
F812: 25 2E            112            AND  MASK             AND MASK
F814: 51 26            113            EOR  (GBASL),Y         XOR DATA
F816: 91 26            114            STA  (GBASL),Y          TO DATA
F818: 60               115            RTS
F819: 20 00 F8         116  HLINE     JSR  PLOT           PLOT SQUARE
F81C: C4 2C            117  HLINE1    CPY  H2             DONE?
F81E: B0 11            118            BCS  RTS1            YES, RETURN
F820: C8               119            INY                  NO, INCR INDEX (X-COORD)
F821: 20 0E F8         120            JSR  PLOT1          PLOT NEXT SQUARE
F824: 90 F6            121            BCC  HLINE1         ALWAYS TAKEN
F826: 69 01            122  VLINEZ    ADC  #$01           NEXT Y-COORD
F828: 48               123  VLINE     PHA                  SAVE ON STACK
F829: 20 00 F8         124            JSR  PLOT            PLOT SQUARE
F82C: 68               125            PLA
F82D: C5 2D            126            CMP  V2             DONE?
F82F: 90 F5            127            BCC  VLINEZ          NO,LOOP.
F831: 60               128  RTS1      RTS
F832: A0 2F            129  CLRSCR    LDY  #$2F           MAX Y, FULL SCRN CLR
F834: D0 02            130            BNE  CLRSC2         ALWAYS TAKEN
F836: A0 27            131  CLRTOP    LDY  #$27           MAX Y, TOP SCRN CLR
F838: 84 2D            132  CLRSC2    STY  V2             STORE AS BOTTOM COORD
                       133                          FOR VLINE CALLS
F83A: A0 27            134            LDY  #$27           RIGHTMOST X-COORD (COLUMN)
F83C: A9 00            135  CLRSC3    LDA  #$0            TOP COORD FOR VLINE CALLS
F83E: 85 30            136            STA  COLOR          CLEAR COLOR (BLACK)
F840: 20 28 F8         137            JSR  VLINE          DRAW VLINE
F843: 88               138            DEY                 NEXT LEFTMOST X-COORD
F844: 10 F6            139            BPL  CLRSC3         LOOP UNTIL DONE.
F846: 60               140            RTS
F847: 48               141  GBASCALC  PHA                 FOR INPUT 000DEFGH
F848: 4A               142            LSR  A
```

```
F849: 29 03     143              AND  #$03
F84B: 09 04     144              ORA  #$04          GENERATE GBASH=000001FG
F84D: 85 27     145              STA  GBASH
F84F: 68        146              PLA                AND GBASL=HDEDE000
F850: 29 18     147              AND  #$18
F852: 90 02     148              BCC  GBCALC
F854: 69 7F     149              ADC  #$7F
F856: 85 26     150  GBCALC      STA  GBASL
F858: 0A        151              ASL  A
F859: 0A        152              ASL  A
F85A: 05 26     153              ORA  GBASL
F85C: 85 26     154              STA  GBASL
F85E: 60        155              RTS
F85F: A5 30     156  NXTCOL      LDA  COLOR         INCREMENT COLOR BY 3
F861: 18        157              CLC
F862: 69 03     158              ADC  #$03
F864: 29 0F     159  SETCOL      AND  #$0F          SETS COLOR=17*A MOD 16
F866: 85 30     160              STA  COLOR
F868: 0A        161              ASL  A             BOTH HALF BYTES OF COLOR EQUAL
F869: 0A        162              ASL  A
F86A: 0A        163              ASL  A
F86B: 0A        164              ASL  A
F86C: 05 30     165              ORA  COLOR
F86E: 85 30     166              STA  COLOR
F870: 60        167              RTS
F871: 4A        168  SCRN        LSR  A             READ SCREEN Y-COORD/2
F872: 08        169              PHP                SAVE LSB (CARRY)
F873: 20 47 F8  170              JSR  GBASCALC      CALC BASE ADDRESS
F876: B1 26     171              LDA  (GBASL),Y     GET BYTE
F878: 28        172              PLP                RESTORE LSB FROM CARRY
F879: 90 04     173  SCRN2       BCC  RTMSKZ        IF EVEN, USE LO H
F87B: 4A        174              LSR  A
F87C: 4A        175              LSR  A
F87D: 4A        176              LSR  A             SHIFT HIGH HALF BYTE DOWN
F87E: 4A        177              LSR  A
F87F: 29 0F     178  RTMSKZ      AND  #$0F          MASK 4-BITS
F881: 60        179              RTS
F882: A6 3A     180  INSDS1      LDX  PCL           PRINT PCL,H
F884: A4 3B     181              LDY  PCH
F886: 20 96 FD  182              JSR  PRYX2
F889: 20 48 F9  183              JSR  PRBLNK        FOLLOWED BY A BLANK
F88C: A1 3A     184              LDA  (PCL,X)       GET OP CODE
F88E: A6        185  INSDS2      TAY
F88F: 4A        186              LSR  A             EVEN/ODD TEST
F890: 90 09     187              BCC  IEVEN
F892: 6A        188              ROR  A             BIT 1 TEST
F893: B0 10     189              BCS  ERR           XXXXXX11 INVALID OP
F895: C9 A2     190              CMP  #$A2
F897: F0 0C     191              BEQ  ERR           OPCODE $89 INVALID
F899: 29 87     192              AND  #$87          MASK BITS
F89B: 4A        193  IEVEN       LSR  A             LSB INTO ARRY FOR L/R TEST
F89C: AA        194              TAX
F89D: BD 62 F9  195              LDA  FMT1,X        GET FORMAT INDEX BYTE
F8A0: 20 79 F8  196              JSR  SCRN2         R/L H-BYTE ON CARRY
F8A3: D0 04     197              BNE  GETFMT
F8A5: A0 80     198  ERR         LDY  #$80          SUBSTITUTE $80 FOR INVALID OPS
F8A7: A9 00     199              LDA  #$0           SET PRINT FORMAT INDEX TO 0
F8A9: AA        200  GETFMT      TAX
F8AA: BD A6 F9  201              LDA  FMT2,X        INDEX INTO PRINT FORMAT TABLE
F8AD: 85 2E     202              STA  FORMAT        SAVE FOR ADR FIELD FORMATTING
F8AF: 29 03     203              AND  #$03          MASK FOR 2-BIT LENGTH
                204  *                (P=1 BYTE, 1=2 BYTE  2=3 BYTE)
F8B1: 85 2F     205              STA  LENGTH
F8B3: 98        206              TYA                OPCODE
F8B4: 29 8F     207              AND  #$8F          MASK FOR 1XXX1010 TEST
F8B6: AA        208              TAX                SAVE IT
F8B7: 98        209              TYA                OPCODE TO A AGAIN
F8B8: A0 03     210              LDY  #$03
F8BA: E0 8A     211              CPX  #$8A
F8BC: F0 0B     212              BEQ  MNNDX3
F8BE: 4A        213  MNNDX1      LSR  A
F8BF: 90 08     214              BCC  MNNDX3        FORM INDEX INTO MNEMONIC TABLE
F8C1: 4A        215              LSR  A
```

```
F8C2:  4A          216  MNNDX2    LSR  A              1) 1XXX1010=>00101XXX
F8C3:  09 20       217             ORA  #$20           2) XXXYYY01=>00111XXX
F8C5:  88          218             DEY                   3) XXXYYY10=>00110XXX
F8C6:  D0 FA       219             BNE  MNNDX2         4) XXXYY100=>00100XXX
F8C8:  C8          220             INY                   5) XXXXX000=>000XXXXX
F8C9:  88          221  MNNDX3    DEY
F8CA:  D0 F2       222             BNE  MNNDX1
F8CC:  60          223             RTS
F8CD:  FF FF FF    224             DFB  $FF,$FF,$FF
F8D0:  20 82 F8    225  INSTDSP    JSR  INSDS1         GEN FMT, LEN BYTES
F8D3:  48          226             PHA                 SAVE MNEMONIC TABLE INDEX
F8D4:  B1 3A       227  PRNTOP     LDA  (PCL),Y
F8D6:  20 DA FD    228             JSR  PRBYTE
F8D9:  A2 01       229             LDX  #$01           PRINT 2 BLANKS
F8DB:  20 4A F9    230  PRNTBL     JSR  PRBL2
F8DE:  C4 2F       231             CPY  LENGTH         PRINT INST (1-3 BYTES)
F8E0:  C8          232             INY                 IN A 12 CHR FIELD
F8E1:  90 F1       233             BCC  PRNTOP
F8E3:  A2 03       234             LDX  #$03           CHAR COUNT FOR MNEMONIC PRINT
F8E5:  C0 04       235             CPY  #$04
F8E7:  90 F2       236             BCC  PRNTBL
F8E9:  68          237             PLA                 RECOVER MNEMONIC INDEX
F8EA:  A8          238             TAY
F8EB:  B9 C0 F9    239             LDA  MNEML,Y
F8EE:  85 2C       240             STA  LMNEM          FETCH 3-CHAR MNEMONIC
F8F0:  B9 00 FA    241             LDA  MNEMR,Y          (PACKED IN 2-BYTES)
F8F3:  85 2D       242             STA  RMNEM
F8F5:  A9 00       243  PRMN1     LDA  #$00
F8F7:  A0 05       244             LDY  #$05
F8F9:  06 2D       245  PRMN2     ASL  RMNEM          SHIFT 5 BITS OF
F8FB:  26 2C       246             ROL  LMNEM            CHARACTER INTO A
F8FD:  2A          247             ROL  A                  (CLEARS CARRY)
F8FE:  88          248             DEY
F8FF:  D0 F8       249             BNE  PRMN2
F901:  69 BF       250             ADC  #$BF           ADD "?" OFFSET
F903:  20 ED FD    251             JSR  COUT           OUTPUT A CHAR OF MNEM
F906:  CA          252             DEX
F907:  D0 EC       253             BNE  PRMN1
F909:  20 48 F9    254             JSR  PRBLNK         OUTPUT 3 BLANKS
F90C:  A4 2F       255             LDY  LENGTH
F90E:  A2 06       256             LDX  #$06           CNT FOR 6 FORMAT BITS
F910:  E0 03       257  PRADR1     CPX  #$03
F912:  F0 1C       258             BEQ  PRADR5         IF X=3 THEN ADDR.
F914:  06 2E       259  PRADR2     ASL  FORMAT
F916:  90 0E       260             BCC  PRADR3
F918:  BD B3 F9    261             LDA  CHAR1-1,X
F91B:  20 ED FD    262             JSR  COUT
F91E:  BD B9 F9    263             LDA  CHAR2-1,X
F921:  F0 03       264             BEQ  PRADR3
F923:  20 ED FD    265             JSR  COUT
F926:  CA          266  PRADR3     DEX
F927:  D0 E7       267             BNE  PRADR1
F929:  60          268             RTS
F92A:  88          269  PRADR4     DEY
F92B:  30 D7       270             BMI  PRADR2
F92D:  20 DA FD    271             JSR  PRBYTE
F930:  A5 2E       272  PRADR5     LDA  FORMAT
F932:  C9 E8       273             CMP  #$E8           HANDLE REL ADR MODE
F934:  B1 3A       274             LDA  (PCL),Y        SPECIAL (PRINT TARGET,
F936:  90 F2       275             BCC  PRADR4           NOT OFFSET)
F938:  20 56 F9    276  RELADR     JSR  PCADJ3
F93B:  AA          277             TAX                 PCL,PCH+OFFSET+1 TO A,Y
F93C:  E8          278             INX
F93D:  D0 01       279             BNE  PRNTYX         +1 TO Y,X
F93F:  C8          280             INY
F940:  98          281  PRNTYX     TYA
F941:  20 DA FD    282  PRNTAX     JSR  PRBYTE         OUTPUT TARGET ADR
F944:  8A          283  PRNTX      TXA                   OF BRANCH AND RETURN
F945:  4C DA FD    284             JMP  PRBYTE
F948:  A2 03       285  PRBLNK     LDX  #$03           BLANK COUNT
F94A:  A9 A0       286  PRBL2      LDA  #$A0           LOAD A SPACE
F94C:  20 ED FD    287  PRBL3      JSR  COUT           OUTPUT A BLANK
F94F:  CA          288             DEX
```

```
F950: D0 F8     289           BNE  PRBL2        LOOP UNTIL COUNT=0
F952: 60        290           RTS
F953: 38        291  PCADJ    SEC               0=1-BYTE,1=2-BYTE,
F954: A5 2F     292  PCADJ2   LDA  LENGTH         2=3-BYTE
F956: A4 3B     293  PCADJ3   LDY  PCH
F958: AA        294           TAX               TEST DISPLACEMENT SIGN
F959: 10 01     295           BPL  PCADJ4         (FOR REL BRANCH)
F95B: 88        296           DEY               EXTEND NEG BY DECR PCH
F95C: 65 3A     297  PCADJ4   ADC  PCL
F95E: 90 01     298           BCC  RTS2         PCL+LENGTH(OR DISPL)+1 TO A
F960: C8        299           INY                 CARRY INTO Y (PCH)
F961: 60        300  RTS2     RTS
                301  *         FMT1 BYTES:         XXXXXXY0 INSTRS
                302  *         IF Y=0              THEN LEFT HALF BYTE
                303  *         IF Y=1              THEN RIGHT HALF BYTE
                304  *                                  (X=INDEX)
F962: 04 20 54
F965: 30 0D     305  FMT1     DFB  $04,$20,$54,$
F967: 80 04 90
F96A: 03 22     306           DFB  $80,$04,$90,$
F96C: 54 33 0D
F96F: 80 04     307           DFB  $54,$33,$0D,$
F971: 90 04 20
F974: 54 33     308           DFB  $90,$04,$20,$
F976: 0D 80 04
F979: 90 04     309           DFB  $0D,$80,$04,$
F97B: 20 54 3B
F97E: 0D 80     310           DFB  $20,$54,$3B,$
F980: 04 90 00
F983: 22 44     311           DFB  $04,$90,$00,$
F985: 33 0D C8
F988: 44 00     312           DFB  $33,$0D,$C8,$
F98A: 11 22 44
F98D: 33 0D     313           DFB  $11,$22,$44,$
F98F: C8 44 A9
F992: 01 22     314           DFB  $C8,$44,$A9,$
F994: 44 33 0D
F997: 80 04     315           DFB  $44,$33,$0D,$
F999: 90 01 22
F99C: 44 33     316           DFB  $90,$01,$22,$
F99E: 0D 80 04
F9A1: 90        317           DFB  $0D,$80,$04,$
F9A2: 26 31 87
F9A5: 9A        318           DFB  $26,$31,$87,$ZZXXXY01 INSTR'S
F9A6: 00        319  FMT2     DFB  $00            ERR
F9A7: 21        320           DFB  $21            IMM
F9A8: 81        321           DFB  $81            Z-PAGE
F9A9: 82        322           DFB  $82            ABS
F9AA: 00        323           DFB  $00            IMPLIED
F9AB: 00        324           DFB  $00            ACCUMULATOR
F9AC: 59        325           DFB  $59            (ZPAG,X)
F9AD: 4D        326           DFB  $4D            (ZPAG),Y
F9AE: 91        327           DFB  $91            ZPAG,X
F9AF: 92        328           DFB  $92            ABS,X
F9B0: 86        329           DFB  $86            ABS,Y
F9B1: 4A        330           DFB  $4A            (ABS)
F9B2: 85        331           DFB  $85            ZPAG,Y
F9B3: 9D        332           DFB  $9D            RELATIVE
F9B4: AC A9 AC
F9B7: A3 A6 A4  333  CHAR1    ASC  ",),#($"
F9BA: D9 00 D8
F9BD: A4 A4 00  334  CHAR2    DFB  $D9,$00,$D8,$
                335  *CHAR2:  "Y",0,"X$$",0
                336  *        MNEML          IS OF FORM:
                337  *        (A)   XXXXX000
                338  *        (B)   XXXYY100
                339  *        (C)   1XXX1010
                340  *        (D)   XXXYYY10
                341  *        (E)   XXXYYY01
                342  *              (X=INDEX)
F9C0: 1C 8A 1C
F9C3: 23 5D 8B  343  MNEML    DFB  $1C,$8A,$1C,$
F9C6: 1B A1 9D
```

```
F9C9:  8A 1D 23 344             DFB   $1B,$A1,$9D,$
F9CC:  9D 8B 1D
F9CF:  A1 00 29 345             DFB   $9D,$8B,$1D,$
F9D2:  19 AE 69
F9D5:  A8 19 23 346             DFB   $19,$AE,$69,$
F9D8:  24 53 1B
F9DB:  23 24 53 347             DFB   $24,$53,$1B,$
F9DE:  19 A1    348             DFB   $19,$A1       (A) FORMAT ABOVE
F9E0:  00 1A 5B
F9E3:  5B A5 69 349             DFB   $00,$1A,$5E,$
F9E6:  24 24    350             DFB   $24,$24       (B) FORMAT
F9E8:  AE AE A8
F9EB:  AD 29 00 351             DFB   $AE,$AE,$A8,$
F9EE:  7C 00    352             DFB   $7C,$00       (C) FORMAT
F9F0:  15 9C 6D
F9F3:  9C A5 69 353             DFB   $15,$9C,$6D,$
F9F6:  29 53    354             DFB   $29,$53       (D) FORMAT
F9F8:  84 13 34
F9FB:  11 A5 69 355             DFB   $84,$13,$34,$
F9FE:  23 A0    356             DFB   $23,$A0       (E) FORMAT
FA00:  D8 62 5A
FA03:  48 26 62 357  MNEMR      DFB   $D8,$62,$5A,$
FA06:  94 88 54
FA09:  44 C8 54 358             DFB   $94,$88,$54,$
FA0C:  68 44 E8
FA0F:  94 00 B4 359             DFB   $68,$44,$E8,$
FA12:  08 84 74
FA15:  B4 28 6E 360             DFB   $08,$84,$74,$
FA18:  74 F4 CC
FA1B:  4A 72 F2 361             DFB   $74,$F4,$CC,$
FA1E:  A4 8A    362             DFB   $A4,$8A       (A) FORMAT
FA20:  00 AA A2
FA23:  A2 74 74 363             DFB   $00,$AA,$A2,$
FA26:  74 72    364             DFB   $74,$72       (B) FORMAT
FA28:  44 68 B2
FA2B:  32 B2 00 365             DFB   $44,$68,$B2,$
FA2E:  22 00    366             DFB   $22,$00       (C) FORMAT
FA30:  1A 1A 26
FA33:  26 72 72 367             DFB   $1A,$1A,$26,$
FA36:  88 C8    368             DFB   $88,$C8       (D) FORMAT
FA38:  C4 CA 26
FA3B:  48 44 44 369             DFB   $C4,$CA,$26,$
FA3E:  A2 C8    370             DFB   $A2,$C8       (E) FORMAT
FA40:  FF FF FF 371             DFB   $FF,$FF,$FF
FA43:  20 D0 F8 372  STEP       JSR   INSTDSP       DISASSEMBLE ONE INST
FA46:  68       373             PLA                   AT (PCL,H)
FA47:  85 2C    374             STA   RTNL          ADJUST TO USER
FA49:  68       375             PLA                   STACK. SAVE
FA4A:  85 2D    376             STA   RTNH            RTN ADR.
FA4C:  A2 08    377             LDX   #$08
FA4E:  BD 10 FB 378  XQINIT     LDA   INITBL-1,X    INIT XEQ AREA
FA51:  95 3C    379             STA   XQT,X
FA53:  CA       380             DEX
FA54:  D0 F8    381             BNE   XQINIT
FA56:  A1 3A    382             LDA   (PCL,X)       USER OPCODE BYTE
FA58:  F0 42    383             BEQ   XBRK          SPECIAL IF BREAK
FA5A:  A4 2F    384             LDY   LENGTH        LEN FROM DISASSEMBLY
FA5C:  C9 20    385             CMP   #$20
FA5E:  F0 59    386             BEQ   XJSR          HANDLE JSR, RTS, JMP,
FA60:  C9 60    387             CMP   #$60            JMP ( ), RTI SPECIAL
FA62:  F0 45    388             BEQ   XRTS
FA64:  C9 4C    389             CMP   #$4C
FA66:  F0 5C    390             BEQ   XJMP
FA68:  C9 6C    391             CMP   #$6C
FA6A:  F0 59    392             BEQ   XJMPAT
FA6C:  C9 40    393             CMP   #$40
FA6E:  F0 55    394             BEQ   XRTI
FA70:  29 1F    395             AND   #$1F
FA72:  49 14    396             EOR   #$14
FA74:  C9 04    397             CMP   #$04          COPY USER INST TO XEQ AREA
FA76:  F0 02    398             BEQ   XQ2             WITH TRAILING NOPS
FA78:  B1 3A    399  XQ1        LDA   (PCL),Y       CHANGE REL BRANCH
FA7A:  99 3C 00 400  XQ2        STA   XQTN2,Y         DISP TO 4 FOR
```

```
FA7D:  88          401
FA7E:  10 F8       402
FA80:  20 3F FF    403
FA83:  4C 3C 00    404
FA86:  85 45       405  IRQ
FA88:  68          406
FA89:  48          407
FA8A:  0A          408
FA8B:  0A          409
FA8C:  0A          410
FA8D:  30 03       411
FA8F:  6C FE 03    412
FA92:  28          413  BREAK
FA93:  20 4C FF    414
FA96:  68          415
FA97:  85 3A       416
FA99:  68          417
FA9A:  85 3B       418
FA9C:  20 82 F8    419  XBRK
FA9F:  20 DA FA    420
FAA2:  4C 65 FF    421
FAA5:  18          422  XRTI
FAA6:  68          423
FAA7:  85 48       424
FAA9:  68          425  XRTS
FAAA:  85 3A       426
FAAC:  68          427
FAAD:  85 3B       428  PCINC2
FAAF:  A5 2F       429  PCINC3
FAB1:  20 56 F9    430
FAB4:  84 3B       431
FAB6:  18          432
FAB7:  90 14       433
FAB9:  18          434  XJSR
FABA:  20 54 F9    435
FABD:  AA          436
FABE:  98          437
FABF:  48          438
FAC0:  8A          439
FAC1:  48          440
FAC2:  A0 02       441
FAC4:  18          442  XJMP
FAC5:  B1 3A       443  XJMPAT
FAC7:  AA          444
FAC8:  88          445
FAC9:  B1 3A       446
FACB:  66 3B       447
FACD:  85 3A       448  NEWPCL
FACF:  B0 F3       449
FAD1:  A5 2D       450  RTNJMP
FAD3:  48          451
FAD4:  A5 2C       452
FAD6:  48          453
FAD7:  20 8E FD    454  REGDSP
FADA:  A9 45       455  RGDSP1
FADC:  85 40       456
FADE:  A9 00       457
FAE0:  85 41       458
FAE2:  A2 FB       459
FAE4:  A9 A0       460  RDSP1
FAE6:  20 ED FD    461
FAE9:  BD 1E FA    462
FAEC:  20 ED FD    463
FAEF:  A9 BD       464
FAF1:  20 ED FD    465
FAF4:  B5 4A       466
FAF6:  20 DA FD    467
FAF9:  E8          468
FAFA:  30 E8       469
FAFC:  60          470
FAFD:  18          471  BRANCH
FAFE:  A0 01       472
       B1 3A       473

DEY                     JMP TO BRANCH OR
BPL  XQ1                NBRANCH FROM XEQ.
JSR  RESTORE            RESTORE USER REG CONTENTS.
JMP  XQTNZ              XEQ USER OP FROM RAM
STA  ACC                  (RETURN TO NBRANCH)
PLA
PHA                     **IRQ HANDLER
ASL  A
ASL  A
ASL  A
BMI  BREAK              TEST FOR BREAK
JMP  (IRQLOC)           USER ROUTINE VECTOR IN RAM
PLP
JSR  SAV1               SAVE REG'S ON BREAK
PLA                       INCLUDING PC
STA  PCL
PLA
STA  PCH
JSR  INSDS1             PRINT USER PC.
JSR  RGDSP1               AND REG'S
JMP  MON                GO TO MONITOR
CLC
PLA                     SIMULATE RTI BY EXPECTING
STA  STATUS               STATUS FROM STACK, THEN RTS
PLA                     RTS SIMULATION
STA  PCL                  EXTRACT PC FROM STACK
PLA                       AND UPDATE PC BY 1 (LEN=0)
STA  PCH
LDA  LENGTH             UPDATE PC BY LEN
JSR  PCADJ3
STY  PCH
CLC
BCC  NEWPCL
CLC
JSR  PCADJ2             UPDATE PC AND PUSH
TAX                       ONTO STACK FOR
TYA                       JSR SIMULATE
PHA
TXA
PHA
LDY  #$02
CLC
LDA  (PCL),Y
TAX                     LOAD PC FOR JMP,
DEY                       (JMP) SIMULATE.
LDA  (PCL),Y
STX  PCH
STA  PCL
BCS  XJMP
LDA  RTNH
PHA
LDA  RTNL
PHA
JSR  CROUT              DISPLAY USER REG
LDA  #ACC                 CONTENTS WITH
STA  A3L                  LABELS
LDA  #ACC/256
STA  A3H
LDX  #$FB
LDA  #$A0
JSR  COUT
LDA  RTBL-$FB,X
JSR  COUT
LDA  #$BD
JSR  COUT
LDA  ACC+5,X
JSR  PRBYTE
INX
BMI  RDSP1
RTS
CLC                     BRANCH TAKEN
LDY  #$01                 ADD LEN+2 TO PC
LDA  (PCL),Y
```

```
FB02:  20 56 F9 474            JSR  PCADJ3
FB05:  85 3A    475            STA  PCL
FB07:  98       476            TYA
FB08:  38       477            SEC
FB09:  B0 A2    478            BCS  PCINC2
FB0B:  20 4A FF 479  NBRNCH    JSR  SAVE        NORMAL RETURN AFTER
FB0E:  38       480                               XEQ USER OP
FB0F:  B0 9E    481            BCS  PCINC3      GO UPDATE PC
FB11:  EA       482  INITBL    NOP
FB12:  EA       483            NOP              DUMMY FILL FOR
FB13:  4C 0B FB 484            JMP  NBRNCH        XEQ AREA
FB16:  4C FD FA 485            JMP  BRANCH
FB19:  C1       486  RTBL      DFB  $C1
FB1A:  D8       487            DFB  $D8
FB1B:  D9       488            DFB  $D9
FB1C:  D0       489            DFB  $D0
FB1D:  D3       490            DFB  $D3
FB1E:  AD 70 C0 491  PREAD     LDA  PTRIG       TRIGGER PADDLES
FB21:  A0 00    492            LDY  #$00        INIT COUNT
FB23:  EA       493            NOP              COMPENSATE FOR 1ST COUNT
FB24:  EA       494            NOP
FB25:  BD 64 C0 495  PREAD2    LDA  PADDL0,X    COUNT Y-REG EVERY
FB28:  10 04    496            BPL  RTS2D         12 USEC
FB2A:  C8       497            INY
FB2B:  D0 F8    498            BNE  PREAD2        EXIT AT 255 MAX
FB2D:  88       499            DEY
FB2E:  60       500  RTS2D     RTS
FB2F:  A9 00    501  INIT      LDA  #$00        CLR STATUS FOR DEBUG
FB31:  85 48    502            STA  STATUS        SOFTWARE
FB33:  AD 56 C0 503            LDA  LORES
FB36:  AD 54 C0 504            LDA  LOWSCR      INIT VIDEO MODE
FB39:  AD 51 C0 505  SETTXT    LDA  TXTSET      SET FOR TEXT MODE
FB3C:  A9 00    506            LDA  #$00          FULL SCREEN WINDOW
FB3E:  F0 0B    507            BEQ  SETWND
FB40:  AD 50 C0 508  SETGR     LDA  TXTCLR      SET FOR GRAPHICS MODE
FB43:  AD 53 C0 509            LDA  MIXSET        LOWER 4 LINES AS
FB46:  20 36 F8 510            JSR  CLRTOP        TEXT WINDOW
FB49:  A9 14    511            LDA  #$14
FB4B:  85 22    512  SETWND    STA  WNDTOP      SET FOR 40 COL WINDOW
FB4D:  A9 00    513            LDA  #$00          TOP IN A-REG,
FB4F:  85 20    514            STA  WNDLFT        BTTM AT LINE 24
FB51:  A9 28    515            LDA  #$28
FB53:  85 21    516            STA  WNDWDTH
FB55:  A9 18    517            LDA  #$18
FB57:  85 23    518            STA  WNDBTM        VTAB TO ROW 23
FB59:  A9 17    519            LDA  #$17
FB5B:  85 25    520  TABV      STA  CV          VTABS TO ROW IN A-REG
FB5D:  4C 22 FC 521            JMP  VTAB
FB60:  20 A4 FB 522  MULPM     JSR  MD1         ABS VAL OF AC AUX
FB63:  A0 10    523  MUL       LDY  #$10        INDEX FOR 16 BITS
FB65:  A5 50    524  MUL2      LDA  ACL         ACX * AUX + XTND
FB67:  4A       525            LSR  A             TO AC, XTND
FB68:  90 0C    526            BCC  MUL4        IF NO CARRY,
FB6A:  18       527            CLC                NO PARTIAL PROD.
FB6B:  A2 FE    528            LDX  #$FE
FB6D:  B5 54    529  MUL3      LDA  XTNDL+2,X   ADD MPLCND (AUX)
FB6F:  75 56    530            ADC  AUXL+2,X     TO PARTIAL PROD
FB71:  95 54    531            STA  XTNDL+2,X      (XTND).
FB73:  E8       532            INX
FB74:  D0 F7    533            BNE  MUL3
FB76:  A2 03    534  MUL4      LDX  #$03
FB78:  76       535  MUL5      DFB  #$76
FB79:  50       536            DFB  #$50
FB7A:  CA       537            DEX
FB7B:  10 FB    538            BPL  MUL5
FB7D:  88       539            DEY
FB7E:  D0 E5    540            BNE  MUL2
FB80:  60       541            RTS
FB81:  20 A4 FB 542  DIVPM     JSR  MD1         ABS VAL OF AC, AUX.
FB84:  A0 10    543  DIV       LDY  #$10        INDEX FOR 16 BITS
FB86:  06 50    544  DIV2      ASL  ACL
FB88:  26 51    545            ROL  ACH
FB8A:  26 52    546            ROL  XTNDL       XTND/AUX
```

```
FB8C:  2A 53     547
FB8E:  38        548
FB8F:  A5 52     549
FB91:  E5 54     550
FB93:  AA        551
FB94:  A5 53     552
FB96:  E5 55     553
FB98:  90 06     554
FB9A:  86 52     555
FB9C:  85 53     556
FB9E:  E6 50     557
FBA0:  88        558  DIV3    DEY
FBA1:  D0 E3     559          BNE   DIV2
FBA3:  60        560          RTS
FBA4:  A0 00     561  MD1     LDY   #$00          ABS VAL OF AC, AUX
FBA6:  84 2F     562          STY   SIGN            WITH RESULT SIGN
FBA8:  A2 54     563          LDX   #AUXL           IN LSB OF SIGN.
FBAA:  20 AF FB  564          JSR   MD2
FBAD:  A2 50     565          LDX   #ACL
FBAF:  B5 01     566  MD2     LDA   LOC1,X        X SPECIFIES AC OR AUX
FBB1:  10 0D     567          BPL   MDRTS
FBB3:  38        568          SEC
FBB4:  98        569  MD3     TYA
FBB5:  F5 00     570          SBC   LOC0,X        COMPL SPECIFIED REG
FBB7:  95 00     571          STA   LOC0,X          IF NEG.
FBB9:  98        572          TYA
FBBA:  F5 01     573          SBC   LOC1,X
FBBC:  95 01     574          STA   LOC1,X
FBBE:  E6 2F     575          INC   SIGN
FBC0:  60        576  MDRTS   RTS
FBC1:  48        577  BASCALC PHA                 CALC BASE ADR IN BASL,H
FBC2:  4A        578          LSR   A                FOR GIVEN LINE NO.
FBC3:  29 03     579          AND   #$03             0<=LINE NO.<=$17
FBC5:  09 04     580          ORA   #$04          ARG=000ABCDE, GENERATE
FBC7:  85 29     581          STA   BASH             BASH=000001CD
FBC9:  68        582          PLA                    AND
FBCA:  29 18     583          AND   #$18             BASL=EABAB000
FBCC:  90 02     584          BCC   BSCLC2
FBCE:  69 7F     585          ADC   #$7F
FBD0:  85 28     586  BSCLC2  STA   BASL
FBD2:  0A        587          ASL   A
FBD3:  0A        588          ASL   A
FBD4:  05 28     589          ORA   BASL
FBD6:  85 28     590          STA   BASL
FBD8:  60        591          RTS
FBD9:  C9 87     592  BELL1   CMP   #$87          BELL CHAR? (CNTRL-G)
FBDB:  D0 12     593          BNE   RTS2B            NO,RETURN
FBDD:  A9 40     594          LDA   #$40          DELAY .01 SECONDS
FBDF:  20 A8 FC  595          JSR   WAIT
FBE2:  A0 C0     596          LDY   #$C0
FBE4:  A9 0C     597  BELL2   LDA   #$0C          TOGGLE SPEAKER AT
FBE6:  20 A8 FC  598          JSR   WAIT             1 KHZ FOR .1 SEC.
FBE9:  AD 30 C0  599          LDA   SPKR
FBEC:  88        600          DEY
FBED:  D0 F5     601          BNE   BELL2
FBEF:  60        602  RTS2B   RTS
FBF0:  A4 24     603  STOADV  LDY   CH            CURSER H INDEX TO Y-REG
FBF2:  91 28     604          STA   (BASL),Y      STOR CHAR IN LINE
FBF4:  E6 24     605  ADVANCE INC   CH            INCR MENT CURSER H INDE
FBF6:  A5 24     606          LDA   CH               (MOVE RIGHT)
FBF8:  C5 21     607          CMP   WNDWDTH       BEYOND WINDOW WIDTH?
FBFA:  B0 66     608          BCS   CR               YES CR TO NEXT LINE
FBFC:  60        609  RTS3    RTS                    NO,RETURN
FBFD:  C9 A0     610  VIDOUT  CMP   #$A0          CONTROL CHAR?
FBFF:  B0 EF     611          BCS   STOADV           NO,OUTPUT IT.
FC01:  A8        612          TAY                 INVERSE VIDEO?
FC02:  10 EC     613          BPL   STOADV           YES, OUTPUT IT.
FC04:  C9 8D     614          CMP   #$8D          CR?
FC06:  F0 5A     615          BEQ   CR               YES.
FC08:  C9 8A     616          CMP   #$8A          LINE FEED?
FC0A:  F0 5A     617          BEQ   LF               IF SO, DO IT.
FC0C:  C9 88     618          CMP   #$88          BACK SPACE? (CNTRL-H)
FC0E:  D0 C9     619          BNE   BELL1            NO, CHECK FOR BELL.
```

```
FC10:  C6 24     620  BS      DEC  CH        DECREMENT CURSER H INDEX
FC12:  10 E8     621          BPL  RTS3      IF POS, OK. ELSE MOVE UP
FC14:  A5 21     622          LDA  WNDWDTH   SET CH TO WNDWDTH-1
FC16:  85 24     623          STA  CH
FC18:  C6 24     624          DEC  CH        (RIGHTMOST SCREEN POS)
FC1A:  A5 22     625  UP      LDA  WNDTOP    CURSER V INDEX
FC1C:  C5 25     626          CMP  CV
FC1E:  B0 0B     627          BCS  RTS4      IF TOP LINE THEN RETURN
FC20:  C6 25     628          DEC  CV        DECR CURSER V-INDEX
FC22:  A5 25     629  VTAB    LDA  CV        GET CURSER V-INDEX
FC24:  20 C1 FB  630  VTABZ   JSR  BASCALC   GENERATE BASE ADDR
FC27:  65 20     631          ADC  WNDLEFT   ADD WINDOW LEFT INDEX
FC29:  85 28     632          STA  BASL      TO BASL
FC2B:  60        633  RTS4    RTS
FC2C:  49 C0     634  ESC1    EOR  #$C0      ESC?
FC2E:  F0 28     635          BEQ  HOME        IF SO, DO HOME AND CLEAR
FC30:  69 FD     636          ADC  #$FD      ESC-A OR B CHECK
FC32:  90 C0     637          BCC  ADVANCE     A, ADVANCE
FC34:  F0 DA     638          BEQ  BS          B, BACKSPACE
FC36:  69 FD     639          ADC  #$FD      ESC-C OR D CHECK
FC38:  90 2C     640          BCC  LF          C, DOWN
FC3A:  F0 DE     641          BEQ  UP          D, GO UP
FC3C:  69 FD     642          ADC  #$FD      ESC-E OR F CHECK
FC3E:  90 5C     643          BCC  CLREOL      E, CLEAR TO END OF LINE
FC40:  D0 E9     644          BNE  RTS4        NOT F, RETURN
FC42:  A4 24     645  CLREOP  LDY  CH        CURSOR H TO Y INDEX
FC44:  A5 25     646          LDA  CV        CURSOR V TO A-REGISTER
FC46:  48        647  CLEOP1  PHA           SAVE CURRENT LINE ON STK
FC47:  20 24 FC  648          JSR  VTABZ     CALC BASE ADDRESS
FC4A:  20 9E FC  649          JSR  CLEOLZ    CLEAR TO EOL, SET CARRY
FC4D:  A0 00     650          LDY  #$00      CLEAR FROM H INDEX=0 FOR REST
FC4F:  68        651          PLA           INCREMENT CURRENT LINE
FC50:  69 00     652          ADC  #$00      (CARRY IS SET)
FC52:  C5 23     653          CMP  WNDBTM    DONE TO BOTTOM OF WINDOW?
FC54:  90 F0     654          BCC  CLEOP1      NO, KEEP CLEARING LINES
FC56:  B0 CA     655          BCS  VTAB        YES, TAB TO CURRENT LINE
FC58:  A5 22     656  HOME    LDA  WNDTOP    INIT CURSOR V
FC5A:  85 25     657          STA  CV          AND H-INDICES
FC5C:  A0 00     658          LDY  #$00
FC5E:  84 24     659          STY  CH        THEN CLEAR TO END OF PAGE
FC60:  F0 E4     660          BEQ  CLEOP1
FC62:  A9 00     661  CR      LDA  #$00      CURSOR TO LEFT OF INDEX
FC64:  85 24     662          STA  CH        (RET CURSOR H=0)
FC66:  E6 25     663  LF      INC  CV        INCR CURSOR V(DOWN 1 LINE)
FC68:  A5 25     664          LDA  CV
FC6A:  C5 23     665          CMP  WNDBTM    OFF SCREEN?
FC6C:  90 B6     666          BCC  VTABZ       NO, SET BASE ADDR
FC6E:  E6 25     667          DEC  CV        DECR CURSOR V(BACK TO BOTTO
FC70:  A5 22     668  SCROLL  LDA  WNDTOP    START AT TOP OF SCRL WNDW
FC72:  48        669          PHA
FC73:  20 24 FC  670          JSR  VTABZ     GENERATE BASE ADDRESS
FC76:  A5 28     671  SCRL1   LDA  BASL      COPY BASL,H
FC78:  85 2A     672          STA  BAS2L       TO BAS2L,H
FC7A:  A5 29     673          LDA  BASH
FC7C:  85 2B     674          STA  BAS2H
FC7E:  A4 21     675          LDY  WNDWDTH   INIT Y TO RIGHTMOST INDEX
FC80:  88        676          DEY              OF SCROLLING WINDOW
FC81:  68        677          PLA
FC82:  69 01     678          ADC  #$01      INCR LINE NUMBER
FC84:  C5 23     679          CMP  WNDBTM    DONE?
FC86:  B0 0D     680          BCS  SCRL3       YES, FINISH
FC88:  48        681          PHA
FC89:  20 24 FC  682          JSR  VTABZ     FORM BASL,H (BASE ADDR)
FC8C:  B1 28     683  SCRL2   LDA  (BASL),Y   MOVE A CHR UP ON LINE
FC8E:  91 2A     684          STA  (BAS2L),Y
FC90:  88        685          DEY            NEXT CHAR OF LINE
FC91:  10 F9     686          BPL  SCRL2
FC93:  30 E1     687          BMI  SCRL1     NEXT LINE
FC95:  A0 00     688  SCRL3   LDY  #$00      CLEAR BOTTOM LINE
FC97:  20 9E FC  689          JSR  CLEOLZ    GET BASE ADDR FOR BOTTOM LINE
FC9A:  B0 B6     690          BCS  VTAB      CARRY IS SET
FC9C:  A4 24     691  CLREOL  LDY  CH        CURSOR H INDEX
FC9E:  A9 A0     692  CLEOLZ  LDA  #$A0
```

```
FCA0:  91 28     693  CLEOL2    STA  (BASL),Y     STORE BLANKS FROM 'HERE'
FCA2:  C8        694               INY              TO END OF LINES (WNDWDTH)
FCA3:  C4 21     695               CPY  WNDWDTH
FCA5:  90 F9     696               BCC  CLEOL2
FCA7:  60        697               RTS
FCA8:  38        698  WAIT        SEC
FCA9:  48        699  WAIT2       PHA
FCAA:  E9 01     700  WAIT3       SBC  #$01        1.0204 USEC
FCAC:  D0 FC     701               BNE  WAIT3       (13+2712*A+512*A*A)
FCAE:  68        702               PLA
FCAF:  E9 01     703               SBC  #$01
FCB1:  D0 F6     704               BNE  WAIT2
FCB3:  60        705               RTS
FCB4:  E6 42     706  NXTA4       INC  A4L         INCR 2-BYTE A4
FCB6:  D0 02     707               BNE  NXTA1       AND A1
FCB8:  E6 43     708               INC  A4H
FCBA:  A5 3C     709  NXTA1       LDA  A1L         INCR 2-BYTE A1.
FCBC:  C5 3E     710               CMP  A2L
FCBE:  A5 3D     711               LDA  A1H         AND COMPARE TO A2
FCC0:  E5 3F     712               SBC  A2H
FCC2:  E6 3C     713               INC  A1L         (CARRY SET IF >=)
FCC4:  D0 02     714               BNE  RTS4B
FCC6:  E6 3D     715               INC  A1H
FCC8:  60        716  RTS4B       RTS
FCC9:  A0 4B     717  HEADR       LDY  #$4B        WRITE A*256 'LONG 1'
FCCB:  20 DB FC  718               JSR  ZERDLY      HALF CYCLES
FCCE:  D0 F9     719               BNE  HEADR       (650 USEC EACH )
FCD0:  69 FE     720               ADC  #$FE
FCD2:  B0 F5     721               BCS  HEADR       THEN A 'SHORT 0'
FCD4:  A0 21     722               LDY  #$21        (400 USEC)
FCD6:  20 DB FC  723  WRBIT       JSR  ZERDLY      WRITE TWO HALF CYCLES
FCD9:  C8        724               INY              OF 250 USEC ('0')
FCDA:  C8        725               INY              OR 500 USEC ('0')
FCDB:  88        726  ZERDLY      DEY
FCDC:  D0 FD     727               BNE  ZERDLY
FCDE:  90 05     728               BCC  WRTAPE      Y IS COUNT FOR
FCE0:  A0 32     729               LDY  #$32        TIMING LOOP
FCE2:  88        730  ONEDLY      DEY
FCE3:  D0 FD     731               BNE  ONEDLY
FCE5:  AC 20 C0  732  WRTAPE      LDY  TAPEOUT
FCE8:  A0 2C     733               LDY  #$2C
FCEA:  CA        734               DEX
FCEB:  60        735               RTS
FCEC:  A2 08     736  RDBYTE      LDX  #$08        8 BITS TO READ
FCEE:  48        737  RDBYT2      PHA              READ TWO TRANSITIONS
FCEF:  20 FA FC  738               JSR  RD2BIT      (FIND EDGE)
FCF2:  68        739               PLA
FCF3:  2A        740               ROL  A           NEXT BIT
FCF4:  A0 3A     741               LDY  #$3A        COUNT FOR SAMPLES
FCF6:  CA        742               DEX
FCF7:  D0 F5     743               BNE  RDBYT2
FCF9:  60        744               RTS
FCFA:  20 FD FC  745  RD2BIT      JSR  RDBIT
FCFD:  88        746  RDBIT       DEY              DECR Y UNTIL
FCFE:  AD 60 C0  747               LDA  TAPEIN      TAPE TRANSITION
FD01:  45 2F     748               EOR  LASTIN
FD03:  10 F8     749               BPL  RDBIT
FD05:  45 2F     750               EOR  LASTIN
FD07:  85 2F     751               STA  LASTIN
FD09:  C0 80     752               CPY  #$80        SET CARRY ON Y-REG.
FD0B:  60        753               RTS
FD0C:  A4 24     754  RDKEY       LDY  CH
FD0E:  B1 28     755               LDA  (BASL),Y    SET SCREEN TO FLASH
FD10:  48        756               PHA
FD11:  29 3F     757               AND  #$3F
FD13:  09 40     758               ORA  #$40
FD15:  91 28     759               STA  (BASL),Y
FD17:  68        760               PLA
FD18:  6C 38 00  761               JMP  (KSWL)      GO TO USER KEY-IN
FD1B:  E6 4E     762  KEYIN       INC  RNDL        INCR RND NUMBER
FD1D:  D0 02     763               BNE  KEYIN2
FD1F:  E6 4F     764               INC  RNDH
FD21:  2C 00 C0  765  KEYIN2      BIT  KBD         KEY DOWN?
```

```
FD24:  10 F5     766              BPL  KEYIN          LOOP
FD26:  91 26     767              STA  (BASL),Y       REPLACE FLASHING SCREEN
FD28:  AD 00 C0  768              LDA  KBD            GET KEYCODE
FD2B:  2C 10 C0  769              BIT  KBDSTRB        CLR KEY STROBE
FD2E:  60        770              RTS
FD2F:  20 0C FC  771  ESC         JSR  RDKEY          GET KEYCODE
FD32:  20 2C FC  772              JSR  ESC1             HANDLE ESC FUNC.
FD35:  20 0C FD  773  RDCHAR      JSR  RDKEY          READ KEY
FD38:  C9 9B     774              CMP  #$9B           ESC?
FD3A:  F0 F3     775              BEQ  ESC              YES, DON'T RETURN
FD3C:  60        776              RTS
FD3D:  A5 32     777  NOTCR       LDA  INVFLG
FD3F:  48        778              PHA
FD40:  A9 FF     779              LDA  #$FF
FD42:  85 32     780              STA  INVFLG         ECHO USER LINE
FD44:  BD 00 02  781              LDA  IN,X             NON INVERSE
FD47:  20 ED FD  782              JSR  COUT
FD4A:  68        783              PLA
FD4B:  85 32     784              STA  INVFLG
FD4D:  BD 00 02  785              LDA  IN,X
FD50:  C9 88     786              CMP  #$88           CHECK FOR EDIT KEYS
FD52:  F0 1D     787              BEQ  BCKSPC           BS, CTRL-X.
FD54:  C9 98     788              CMP  #$98
FD56:  F0 0A     789              BEQ  CANCEL
FD58:  E0 F8     790              CPX  #$F8           MARGIN?
FD5A:  90 03     791              BCC  NOTCR1
FD5C:  20 3A FF  792              JSR  BELL             YES, SOUND BELL
FD5F:  E8        793  NOTCR1      INX                 ADVANCE INPUT INDEX
FD60:  D0 13     794              BNE  NXTCHAR
FD62:  A9 DC     795  CANCEL      LDA  #$DC           BACKSLASH AFTER CANCELLED LIN
FD64:  20 ED FD  796              JSR  COUT
FD67:  20 8E FD  797  GETLNZ      JSR  CROUT          OUTPUT CR
FD6A:  A5 33     798  GETLN       LDA  PROMPT
FD6C:  20 ED FD  799              JSR  COUT           OUTPUT PROMPT CHAR
FD6F:  A2 01     800              LDX  #$01           INIT INPUT INDEX
FD71:  8A        801  BCKSPC      TXA                   WILL BACKSPACE TO 0
FD72:  F0 F3     802              BEQ  GETLNZ
FD74:  CA        803              DEX
FD75:  20 35 FD  804  NXTCHAR     JSR  RDCHAR
FD78:  C9 95     805              CMP  #PICK          USE SCREEN CHAR
FD7A:  D0 02     806              BNE  CAPTST           FOR CTRL-U
FD7C:  B1 28     807              LDA  (BASL),Y
FD7E:  C9 E0     808  CAPTST     CMP  #$E0
FD80:  90 02     809              BCC  ADDINP         CONVERT TO CAPS
FD82:  29 DF     810              AND  #$DF
FD84:  9D 00 02  811  ADDINP     STA  IN,X           ADD TO INPUT BUF
FD87:  C9 8D     812              CMP  #$8D
FD89:  D0 B2     813              BNE  NOTCR
FD8B:  20 9C FC  814              JSR  CLREOL         CLR TO EOL IF CR
FD8E:  A9 8D     815  CROUT       LDA  #$8D
FD90:  D0 5B     816              BNE  COUT
FD92:  A4 3D     817  PRA1        LDY  A1H            PRINT CR,A1 IN HEX
FD94:  A6 3C     818              LDX  A1L
FD96:  20 8E FD  819  PRYX2      JSR  CROUT
FD99:  20 40 F9  820              JSR  PRNTYX
FD9C:  A0 00     821              LDY  #$00
FD9E:  A9 AD     822              LDA  #$AD           PRINT '-'
FDA0:  4C ED FD  823              JMP  COUT
FDA3:  A5 3C     824  XAM8        LDA  A1L
FDA5:  09 07     825              ORA  #$07           SET TO FINISH AT
FDA7:  85 3E     826              STA  A2L              MOD 8=7
FDA9:  A5 3D     827              LDA  A1H
FDAB:  85 3F     828              STA  A2H
FDAD:  A5 3C     829  MOD8CHK     LDA  A1L
FDAF:  29 07     830              AND  #$07
FDB1:  D0 03     831              BNE  DATAOUT
FDB3:  20 92 FD  832  XAM         JSR  PRA1
FDB6:  A9 A0     833  DATAOUT     LDA  #$A0
FDB8:  20 ED FD  834              JSR  COUT           OUTPUT BLANK
FDBB:  B1 3C     835              LDA  (A1L),Y
FDBD:  20 DA FD  836              JSR  PRBYTE         OUTPUT BYTE IN HEX
FDC0:  20 BA FC  837              JSR  NXTA1
```

```
FDC5:  90 E6     838
FDC5:            839  RTS4C     RTS
FDC6:  4A        840  XAMPM     LSR  A
FDC7:  90 EA     841            BCC  XAM
FDC9:  4A        842            LSR  A
FDCA:  4A        843            LSR  A
FDCB:  A5 3E     844            LDA  A2L
FDCD:  90 02     845            BCC  ADD
FDCF:  49 FF     846            EOR  #$FF
FDD1:  65 3C     847  ADD       ADC  A1L
FDD3:  48        848            PHA
FDD4:  A9 BD     849            LDA  #$BD
FDD6:  20 ED FD  850            JSR  COUT
FDD9:  68        851            PLA
FDDA:  48        852  PRBYTE    PHA
FDDB:  4A        853            LSR  A
FDDC:  4A        854            LSR  A
FDDD:  4A        855            LSR  A
FDDE:  4A        856            LSR  A
FDDF:  20 E5 FD  857            JSR  PRHEXZ
FDE2:  68        858            PLA
FDE3:  29 0F     859  PRHEX     AND  #$0F
FDE5:  09 B0     860  PRHEXZ    ORA  #$B0
FDE7:  C9 BA     861            CMP  #$BA
FDE9:  90 02     862            BCC  COUT
FDEB:  69 06     863            ADC  #$06
FDED:  6C 36 00  864  COUT      JMP  (CSWL)
FDF0:  C9 A0     865  COUT1     CMP  #$A0
FDF2:  90 02     866            BCC  COUTZ
FDF4:  25 32     867            AND  INVFLG
FDF6:  84 35     868  COUTZ     STY  YSAV1
FDF8:  48        869            PHA
FDF9:  20 FD FB  870            JSR  VIDOUT
FDFC:  68        871            PLA
FDFD:  A4 35     872            LDY  YSAV1
FDFF:  60        873            RTS
FE00:  C6 34     874  BL1       DEC  YSAV
FE02:  F0 9F     875            BEQ  XAM8
FE04:  CA        876  BLANK     DEX
FE05:  D0 16     877            BNE  SETMDZ
FE07:  C9 BA     878            CMP  #$BA
FE09:  D0 BB     879            BNE  XAMPM
FE0B:  85 31     880  STOR      STA  MODE
FE0D:  A5 3E     881            LDA  A2L
FE0F:  91 40     882            STA  (A3L),Y
FE11:  E6 40     883            INC  A3L
FE13:  D0 02     884            BNE  RTS5
FE15:  E6 41     885            INC  A3H
FE17:  60        886  RTS5      RTS
FE18:  A4 34     887  SETMODE   LDY  YSAV
FE1A:  B9 FF 01  888            LDA  IN-1,Y
FE1D:  85 31     889  SETMDZ    STA  MODE
FE1F:  60        890            RTS
FE20:  A2 01     891  LT        LDX  #$01
FE22:  B5 3E     892  LT2       LDA  A2L,X
FE24:  95 42     893            STA  A4L,X
FE26:  95 44     894            STA  A5L,X
FE28:  CA        895            DEX
FE29:  10 F7     896            BPL  LT2
FE2B:  60        897            RTS
FE2C:  B1 3C     898  MOVE      LDA  (A1L),Y
FE2E:  91 42     899            STA  (A4L),Y
FE30:  20 B4 FC  900            JSR  NXTA4
FE33:  90 F7     901            BCC  MOVE
FE35:  60        902            RTS
FE36:  B1 3C     903  VFY       LDA  (A1L),Y
FE38:  D1 42     904            CMP  (A4L),Y
FE3A:  F0 1C     905            BEQ  VFYOK
FE3C:  20 92 FD  906            JSR  PRA1
FE3F:  B1 3C     907            LDA  (A1L),Y
FE41:  20 DA FD  908            JSR  PRBYTE
FE44:  A9 A0     909            LDA  #$A0
FE46:  20 ED FD  910            JSR  COUT

                                BCC  MOD8CHK    CHECK IF TIME TO,
                                RTS               PRINT ADDR
                                LSR  A          DETERMINE IF MON
                                BCC  XAM          MODE IS XAM
                                LSR  A            ADD, OR SUB
                                LSR  A
                                LDA  A2L
                                BCC  ADD
                                EOR  #$FF       SUB: FORM 2'S COMPLEMENT
                                ADC  A1L
                                PHA
                                LDA  #$BD
                                JSR  COUT       PRINT '=', THEN RESULT
                                PLA
                                PHA             PRINT BYTE AS 2 HEX
                                LSR  A            DIGITS, DESTROYS A-REG
                                LSR  A
                                LSR  A
                                LSR  A
                                LSR  A
                                JSR  PRHEXZ
                                PLA
                                AND  #$0F       PRINT HEX DIG IN A-REG
                                ORA  #$B0         LSB'S
                                CMP  #$BA
                                BCC  COUT
                                ADC  #$06
                                JMP  (CSWL)     VECTOR TO USER OUTPUT ROUTINE
                                CMP  #$A0       DON'T OUTPUT CTRL'S INVERSE
                                BCC  COUTZ      MASK WITH INVERSE FLAG
                                AND  INVFLG
                                STY  YSAV1      SAV Y-REG
                                PHA             SAV A-REG
                                JSR  VIDOUT     OUTPUT A-REG AS ASCII
                                PLA             RESTORE A-REG
                                LDY  YSAV1        AND Y-REG
                                RTS               THEN RETURN
                                DEC  YSAV
                                BEQ  XAM8
                                DEX             BLANK TO MON
                                BNE  SETMDZ     AFTER BLANK
                                CMP  #$BA       DATA STORE MODE?
                                BNE  XAMPM        NO, XAM, ADD OR SUB
                                STA  MODE       KEEP IN STORE MODE
                                LDA  A2L
                                STA  (A3L),Y    STORE AS LOW BYTE AS (A3)
                                INC  A3L
                                BNE  RTS5       INCR A3, RETURN
                                INC  A3H
                                RTS
                                LDY  YSAV       SAVE CONVERTED ':', '+',
                                LDA  IN-1,Y       '-', '.' AS MODE.
                                STA  MODE
                                RTS
                                LDX  #$01
                                LDA  A2L,X      COPY A2 (2 BYTES) TO
                                STA  A4L,X        A4 AND A5
                                STA  A5L,X
                                DEX
                                BPL  LT2
                                RTS
                                LDA  (A1L),Y    MOVE (A1 TO A2) TO
                                STA  (A4L),Y      (A4)
                                JSR  NXTA4
                                BCC  MOVE
                                RTS
                                LDA  (A1L),Y    VERIFY (A1 TO A2) WITH
                                CMP  (A4L),Y      (A4)
                                BEQ  VFYOK
                                JSR  PRA1
                                LDA  (A1L),Y
                                JSR  PRBYTE
                                LDA  #$A0
                                JSR  COUT
```

```
FE49:  A9 A8     911            LDA  #$A8
FE4B:  20 ED FD  912            JSR  COUT
FE4E:  B1 42     913            LDA  (A4L),Y
FE50:  20 DA FD  914            JSR  PRBYTE
FE53:  A9 A9     915            LDA  #$A9
FE55:  20 ED FD  916            JSR  COUT
FE58:  20 B4 FC  917  VFYOK     JSR  NXTA4
FE5B:  90 D9     918            BCC  VFY
FE5D:  60        919            RTS
FE5E:  20 75 FE  920  LIST      JSR  A1PC          MOVE A1 (2 BYTES) TO
FE61:  A9 14     921                                 PC IF SPEC'D AND
FE63:  48        922  LIST2     PHA                   DISSEMBLE 20 INSTRS
FE64:  20 D0 F8  923            JSR  INSTDSP
FE67:  20 53 F9  924            JSR  PCADJ         ADJUST PC EACH INSTR
FE6A:  85 3A     925            STA  PCL
FE6C:  84 3B     926            STY  PCH
FE6E:  68        927            PLA
FE6F:  38        928            SEC
FE70:  E9 01     929            SBC  #$01          NEXT OF 20 INSTRS
FE72:  D0 EF     930            BNE  LIST2
FE74:  60        931            RTS
FE75:  0A        932  A1PC      TAX                IF USER SPEC'D ADR
FE76:  F0 07     933            BEQ  A1PCRTS         COPY FROM A1 TO PC
FE78:  B5 3C     934  A1PCLP    LDA  A1L,X
FE7A:  95 3A     935            STA  PCL,X
FE7C:  CA        936            DEX
FE7D:  10 F9     937            BPL  A1PCLP
FE7F:  60        938  A1PCRTS   RTS
FE80:  A0 3F     939  SETINV    LDY  #$3F          SET FOR INVERSE VID
FE82:  D0 02     940            BNE  SETIFLG         VIA COUT1
FE84:  A0 FF     941  SETNORM   LDY  #$FF          SET FOR NORMAL VID
FE86:  84 32     942  SETIFLG   STY  INVFLG
FE88:  60        943            RTS
FE89:  A9 00     944  SETKBD    LDA  #$00          SIMULATE PORT #0 INPUT
FE8B:  85 3E     945  INPORT    STA  A2L             SPECIFIED (KEYIN ROUTINE)
FE8D:  A2 38     946  INPRT     LDX  #KSWL
FE8F:  A0 1B     947            LDY  #KEYIN
FE91:  D0 08     948            BNE  IOPRT
FE93:  A9 00     949  SETVID    LDA  #$00          SIMULATE PORT #0 OUTPUT
FE95:  85 3E     950  OUTPORT   STA  A2L             SPECIFIED (COUT1 ROUTINE)
FE97:  A2 36     951  OUTPRT    LDX  #CSWL
FE99:  A0 F0     952            LDY  #COUT1
FE9B:  A5 3E     953  IOPRT     LDA  A2L           SET RAM IN/OUT VECTORS
FE9D:  29 0F     954            AND  #$0F
FE9F:  F0 06     955            BEQ  IOPRT1
FEA1:  09 C0     956            ORA  #IOADR/256
FEA3:  A0 00     957            LDY  #$00
FEA5:  F0 02     958            BEQ  IOPRT2
FEA7:  A9 FD     959  IOPRT1    LDA  #COUT1/256
FEA9:  94 00     960  IOPRT2    STY  LOC0,X
FEAB:  95 01     961            STA  LOC1,X
FEAD:  60        962            RTS
FEAE:  EA        963            NOP
FEAF:  EA        964            NOP
FEB0:  4C 00 E0  965  XBASIC    JMP  BASIC         TO BASIC WITH SCRATCH
FEB3:  4C 03 E0  966  BASCONT   JMP  BASIC2        CONTINUE BASIC
FEB6:  20 75 FE  967  GO        JSR  A1PC          ADR TO PC IF SPEC'D
FEB9:  20 3F FF  968            JSR  RESTORE       RESTORE META REGS
FEBC:  6C 3A 00  969            JMP  (PCL)         GO TO USER SUBR
FEBF:  4C D7 FA  970  REGZ      JMP  REGDSP        TO REG DISPLAY
FEC2:  C6 34     971  TRACE     DEC  YSAV
FEC4:  20 75 FE  972  STEPZ     JSR  A1PC          ADR TO PC IF SPEC'D
FEC7:  4C 4B FA  973            JMP  STEP          TAKE ONE STEP
FECA:  4C F8 03  974  USR       JMP  USRADR        TO USR SUBR AT USRADR
FECD:  A9 40     975  WRITE     LDA  #$40          WRITE 10-SEC HEADER
FECF:  20 C9 FC  976            JSR  HEADR
FED2:  A0 27     977            LDY  #$27
FED4:  A2 00     978  WR1       LDX  #$00
FED6:  41 3C     979            EOR  (A1L,X)
FED8:  48        980            PHA
FED9:  A1 3C     981            LDA  (A1L,X)
```

```
FEDB:  20 ED FD  982              JSR  WRBYTE
FEDE:  20 BA FC  983              JSR  NXTA1
FEE1:  A0 1D     984              LDY  #$1D
FEE3:  68        985              PLA
FEE4:  90 EE     986              BCC  WR1
FEE6:  A0 22     987              LDY  #$22
FEE8:  20 ED FE  988              JSR  WRBYTE
FEEB:  F0 4D     989              BEQ  BELL
FEED:  A2 10     990  WRBYTE     LDX  #$10
FEEF:  0A        991  WRBYT2     ASL  A
FEF0:  20 D6 FC  992              JSR  WRBIT
FEF3:  D0 FA     993              BNE  WRBYT2
FEF5:  60        994              RTS
FEF6:  20 00 FE  995  CRMON       JSR  BL1          HANDLE CR AS BLANK
FEF9:  68        996                                  THEN POP STACK
FEFA:  68        997                                  AND RTN TO MON
FEFB:  D0 6C     998              BNE  MONZ
FEFD:  20 FA FC  999  READ        JSR  RD2BIT       FIND TAPEIN EDGE
FF00:  A9 16     1000             LDA  #$16
FF02:  20 C9 FC  1001             JSR  HEADR        DELAY 3.5 SECONDS
FF05:  85 2E     1002             STA  CHKSUM       INIT CHKSUM=$FF
FF07:  20 FA FC  1003             JSR  RD2BIT       FIND TAPEIN EDGE
FF0A:  A0 24     1004 RD2         LDY  #$24         LOOK FOR SYNC BIT
FF0C:  20 FD FC  1005             JSR  RDBIT          (SHORT 0)
FF0F:  B0 F9     1006             BCS  RD2            LOOP UNTIL FOUND
FF11:  20 FD FC  1007             JSR  RDBIT        SKIP SECOND SYNC H-CYCLE
FF14:  A0 3B     1008             LDY  #$3B         INDEX FOR 0/1 TEST
FF16:  20 EC FC  1009 RD3         JSR  RDBYTE       READ A BYTE
FF19:  81 3C     1010             STA  (A1L,X)      STORE AT (A1)
FF1B:  45 2E     1011             EOR  CHKSUM
FF1D:  85 2E     1012             STA  CHKSUM       UPDATE RUNNING CHKSUM
FF1F:  20 BA FC  1013             JSR  NXTA1        INCR A1, COMPARE TO A2
FF22:  A0 35     1014             LDY  #$35         COMPENSATE 0/1 INDEX
FF24:  90 F0     1015             BCC  RD3          LOOP UNTIL DONE
FF26:  20 EC FC  1016             JSR  RDBYTE       READ CHKSUM BYTE
FF29:  C5 2E     1017             CMP  CHKSUM
FF2B:  F0 0D     1018             BEQ  BELL         GOOD, SOUND BELL AND RETURN
FF2D:  A9 C5     1019 PRERR       LDA  #$C5
FF2F:  20 ED FD  1020             JSR  COUT         PRINT "ERR", THEN BELL
FF32:  A9 D2     1021             LDA  #$D2
FF34:  20 ED FD  1022             JSR  COUT
FF37:  20 ED FD  1023             JSR  COUT
FF3A:  A9 87     1024 BELL        LDA  #$87         OUTPUT BELL AND RETURN
FF3C:  4C ED FD  1025             JMP  COUT
FF3F:  A5 48     1026 RESTORE     LDA  STATUS       RESTORE 6502 REG CONTENTS
FF41:  48        1027                                 USED BY DEBUG SOFTWARE
FF42:  A5 45     1028             LDA  ACC
FF44:  A6 46     1029 RESTR1      LDX  XREG
FF46:  A4 47     1030             LDY  YREG
FF48:  28        1031             PLP
FF49:  60        1032             RTS
FF4A:  85 45     1033 SAVE        STA  ACC          SAVE 6502 REG CONTENTS
FF4C:  86 46     1034 SAV1        STX  XREG
FF4E:  84 47     1035             STY  YREG
FF50:  08        1036             PHP
FF51:  68        1037             PLA
FF52:  85 48     1038             STA  STATUS
FF54:  BA        1039             TSX
FF55:  86 49     1040             STX  SPNT
FF57:  D8        1041             CLD
FF58:  60        1042             RTS
FF59:  20 84 FE  1043 RESET       JSR  SETNORM      SET SCREEN MODE,
FF5C:  20 2F FB  1044             JSR  INIT           AND INIT KBD/SCREEN
FF5F:  20 93 FE  1045             JSR  SETVID         AS I/O DEV'S
FF62:  20 89 FE  1046             JSR  SETKBD
FF65:  D8        1047 MON         CLD               MUST SET HEX MODE!
FF66:  20 3A FF  1048             JSR  BELL
FF69:  A9 AA     1049 MONZ        LDA  #$AA         '*' PROMPT FOR MON
FF6B:  85 33     1050             STA  PROMPT
FF6D:  20 67 FD  1051             JSR  GETLNZ       READ A LINE
FF70:  20 C7 FF  1052             JSR  ZMODE        CLEAR MON MODE, SCAN IDX
FF73:  20 A7 FF  1053 NXTITM      JSR  GETNUM       GET ITEM, NON-HEX
FF76:  84 34     1054             STY  YSAV           CHAR IN A-REG
```

```
FF78: A0 17     1055          LDY  #$17          X-REG=0 IF NO HEX INPUT
FF7A: 88        1056 CHRSRCH  DEY
FF7B: 30 E6     1057          BMI  MON           NOT FOUND, GO TO MON
FF7D: D9 CC FF  1058          CMP  CHRTBL,Y      FIND CMND CHAR IN TEL
FF80: D0 F8     1059          BNE  CHRSRCH
FF82: 20 BE FF  1060          JSR  TOSUB         FOUND, CALL CORRESPONDING
FF85: A4 34     1061          LDY  YSAV            SUBROUTINE
FF87: 4C 73 FF  1062          JMP  NXTITM
FF8A: A2 03     1063 DIG      LDX  #$03
FF8C: 0A        1064          ASL  A
FF8D: 0A        1065          ASL  A             GOT HEX DIG,
FF8E: 0A        1066          ASL  A               SHIFT INTO A2
FF8F: 0A        1067          ASL  A
FF90: 0A        1068 NXTBIT   ASL  A
FF91: 26 3E     1069          ROL  A2L
FF93: 26 3F     1070          ROL  A2H
FF95: CA        1071          DEX                LEAVE X=$FF IF DIG
FF96: 10 F6     1072          BPL  NXTBIT
FF98: A5 31     1073 NXTBAS   LDA  MODE
FF9A: D0 06     1074          BNE  NXTBS2        IF MODE IS ZERO
FF9C: B5 3F     1075          LDA  A2H,X           THEN COPY A2 TO
FF9E: 95 3D     1076          STA  A1H,X           A1 AND A3
FFA0: 95 41     1077          STA  A3H,X
FFA2: E8        1078 NXTBS2   INX
FFA3: F0 F3     1079          BEQ  NXTBAS
FFA5: D0 06     1080          BNE  NXTCHR
FFA7: A2 00     1081 GETNUM   LDX  #$00          CLEAR A2
FFA9: 86 3E     1082          STX  A2L
FFAB: 86 3F     1083          STX  A2H
FFAD: B5 00 02  1084 NXTCHR   LDA  IN,Y          GET CHAR
FFB0: C8        1085          INY
FFB1: 49 B0     1086          EOR  #$B0
FFB3: C9 0A     1087          CMP  #$0A
FFB5: 90 D3     1088          BCC  DIG           IF HEX DIG, THEN
FFB7: 69 88     1089          ADC  #$88
FFB9: C9 FA     1090          CMP  #$FA
FFBB: B0 CD     1091          BCS  DIG
FFBD: 60        1092          RTS
FFBE: A9 FE     1093 TOSUB    LDA  #GO/256       PUSH HIGH-ORDER
FFC0: 48        1094          PHA                  SUBR ADR ON STK
FFC1: B9 E3 FF  1095          LDA  SUBTBL,Y      PUSH LOW ORDER
FFC4: 48        1096          PHA                  SUBR ADR ON STK
FFC5: A5 31     1097          LDA  MODE
FFC7: A0 00     1098 ZMODE    LDY  #$00          CLR MODE, OLD MODE
FFC9: 84 31     1099          STY  MODE            TO A-REG
FFCB: 60        1100          RTS                GO TO SUBR VIA RTS
FFCC: BC        1101 CHRTBL   DFB  $BC           F("CTRL-C")
FFCD: B2        1102          DFB  $B2           F("CTRL-Y")
FFCE: BE        1103          DFB  $BE           F("CTRL-E")
FFCF: ED        1104          DFB  $ED           F("T")
FFD0: EF        1105          DFB  $EF           F("V")
FFD1: C4        1106          DFB  $C4           F("CTRL-K")
FFD2: EC        1107          DFB  $EC           F("S")
FFD3: A9        1108          DFB  $A9           F("CTRL-P")
FFD4: BB        1109          DFB  $BB           F("CTRL-B")
FFD5: A6        1110          DFB  $A6           F("-")
FFD6: A4        1111          DFB  $A4           F("+")
FFD7: 06        1112          DFB  $06           F("M") (F=EX-OR $B0+$89)
FFD8: 95        1113          DFB  $95           F("<")
FFD9: 07        1114          DFB  $07           F("N")
FFDA: 02        1115          DFB  $02           F("I")
FFDB: 05        1116          DFB  $05           F("L")
FFDC: F0        1117          DFB  $F0           F("W")
FFDD: 00        1118          DFB  $00           F("G")
FFDE: EB        1119          DFB  $EB           F("R")
FFDF: 93        1120          DFB  $93           F(":")
FFE0: A7        1121          DFB  $A7           F(".")
FFE1: C6        1122          DFB  $C6           F("CR")
FFE2: 99        1123          DFB  $99           F(BLANK)
FFE3: B2        1124 SUBTBL   DFB  #BASCONT-1
FFE4: C9        1125          DFB  #USR-1
FFE5: BE        1126          DFB  #REGZ-1
```

```
FFE6:  C1          1127              DFB  #TRACE-1
FFE7:  35          1128              DFB  #VFY-1
FFE8:  0C          1129              DFB  #INPRT-1
FFE9:  C3          1130              DFB  #STEPZ-1
FFEA:  96          1131              DFB  #OUTPRT-1
FFEB:  AF          1132              DFB  #XBASIC-1
FFEC:  17          1133              DFB  #SETMODE-1
FFED:  17          1134              DFB  #SETMODE-1
FFEE:  2B          1135              DFB  #MOVE-1
FFEF:  1F          1136              DFB  #LT-1
FFF0:  83          1137              DFB  #SETNORM-1
FFF1:  7F          1138              DFB  #SETINV-1
FFF2:  5D          1139              DFB  #LIST-1
FFF3:  CC          1140              DFB  #WRITE-1
FFF4:  B5          1141              DFB  #GO-1
FFF5:  FC          1142              DFB  #READ-1
FFF6:  17          1143              DFB  #SETMODE-1
FFF7:  17          1144              DFB  #SETMODE-1
FFF8:  F5          1145              DFB  #CRMON-1
FFF9:  03          1146              DFB  #BLANK-1
FFFA:  FB          1147              DFB  #NMI          NMI VECTOR
FFFB:  03          1148              DFB  #NMI/256
FFFC:  59          1149              DFB  #RESET        RESET VECTOR
FFFD:  FF          1150              DFB  #RESET/256
FFFE:  86          1151              DFB  #IRQ          IRQ VECTOR
FFFF:  FA          1152              DFB  #IRQ/256
                   1153 XQTNZ        EQU  $3C
```

SYMBOL TABLE
(NUMERICAL ORDER)

0000 LOC0
0022 WNDTOP
0026 GBASL
002A BAS2L
002D V2
002E FORMAT
0030 COLOR
0034 YSAV
0038 KSWL
003C A1L
0040 A3L
0044 A5L
0047 YREG
004F RNDH
03F2 SOFTEV
03FB NMI
C000 IOADR
C030 SPKR
C053 MIXSET
C057 HIRES
C05B CLRAN1
C05F CLRAN3
CFFF CLRROM
F80C RTMASK
F826 VLINEZ
F836 CLRTOP
F856 GBCALC
F87F RTMSKZ
F8A5 ERR
F8C9 MNNDX3
F8F5 NXTCOL
F926 PRADR3
F940 PRNTYX
F94A PRBL2
F956 PCADJ3
F9A6 FMT2
FA00 MNEMR
FA62 RESET
FAA3 NOFIX
FABA SLOOP
FAE4 RDSP1
FB11 XLTBL
FB2E RTS2D
FB4B SETWND
FB6F SETPWRC
FB97 ESCOLD
FBD0 BASCLC2
FBF0 STORADV
FC10 BS
FC2B RTS4
FC58 HOME
FC76 SCRL1
FC9E CLEOLZ
FCAA WAIT3
FCC9 HEADR
FCE5 WRTAPE
FCFD RDBIT
FD2F ESC
FD62 CANCEL
0001 LOC1
0023 WNDBTM
0027 GBASH
002B BAS2H
002D RMNEM
002F LASTIN
0031 MODE
0035 YSAV1
0039 KSWH
003D A1H
0041 A3H
0045 A5H
0048 STATUS
0095 PICK
03F4 PWREDUP
03FE IRQLOC
C000 KBD
C050 TXTCLR
C054 LOWSCR
C058 SETAN0
C05C SETAN2
C060 TAPEIN
E000 BASIC
F80E PLOT1
F828 VLINE
F838 CLRSC2
F864 SETCOL
F882 INSDS1
F8A9 GETFMT
F8D0 INSTDSP
F8F9 PRMN2
F92A PRADR4
F941 PRNTAX
F94C PRBL3
F95C PCADJ4
F9B4 CHAR1
FA40 IRQ
FA6F INITAN
FAA6 PWRUP
FAC7 NXTBYT
FAFD PWRCON
FB19 RTBL
FB2F INIT
FB5B TABV
FD78 VIDWAIT
FB9B ESCNOW
FBD9 BELL1
FBF4 ADVANCE
FC1A UP
FC2C ESC1
FC62 CR
FC8C SCRL2
FCA0 CLEOL2
FCB4 NXTA4
FCD6 WRBIT
FCEC RDBYTE
FD0C RDKEY
FD35 RDCHAR
FD67 GETLNZ
0020 WNDLFT
0024 CH
0028 BASL
002C H2
002E MASK
002F LENGTH
0032 INVFLG
0036 CSWL
003A PCL
003E A2L
0042 A4L
0045 ACC
0049 SPNT
0200 IN
03F5 AMPERV
0400 LINE1
C010 KBDSTRB
C051 TXTSET
C055 HISCR
C059 CLRAN0
C05D CLRAN2
C064 PADDL0
E003 BASIC2
F819 HLINE
F831 RTS1
F83C CLRSC3
F871 SCRN
F88C INSDS2
F8BE MNNDX1
F8D4 PRNTOP
F910 PRADR1
F930 PRADR5
F944 PRNTX
F953 PCADJ
F961 RTS2

```
F9BA CHAR2
FA4C BREAK
FA81 NEWMON
FAA9 SETPG3
FAD7 REGDSP
FB02 DISKID
FB1E PREAD
FB39 SETTXT
FB60 APPLEII
FB88 KBDWAIT
FBA5 ESCNEW
FBE4 BELL2
FBFC RTS3
FC22 VTAB
FC42 CLREOP
FC66 LF
FC95 SCRL3
FCA8 WAIT
FCBA NXTA1
FCDB ZERDLY
FCEE RDBYT2
FD1B KEYIN
FD3D NOTCR
FD6A GETLN
0021 WNDWDTH
0025 CV
0029 BASH
002C LMNEM
002E CHKSUM
002F SIGN
0033 PROMPT
0037 CSWH
003B PCH
003F A2H
0043 A4H
0046 XREG
004E RNDL
03F0 BRKV
03F8 USRADR
07F8 MSLOT
C020 TAPEOUT
C052 MIXCLR
C056 LORES
C05A SETAN1
C05E SETAN3
C070 PTRIG
F800 PLOT
F81C HLINE1
F832 CLRSCR
F847 GBASCALC
F879 SCRN2
F89B IEVEN
F8C2 MNNDX2
F8DB PRNTBL
F914 PRADR2
F938 RELADR
F948 PRBLNK
F954 PCADJ2
F962 FMT1
F9C0 MNEML
FA59 OLDBRK
FA9B FIXSEV
FAAB SETPLP
FADA RGDSP1
FB09 TITLE
FB25 PREAD2
FB40 SETGR
FB65 STITLE
FB94 NOWAIT
FBC1 BASCALC
FBEF RTS2B
FBFD VIDOUT
FC24 VTABZ
FC46 CLEOP1
FC70 SCROLL
FC9C CLREOL
FCA9 WAIT2
FCC8 RTS4B
FCE2 ONEDLY
FCFA RD2BIT
FD21 KEYIN2
FD5F NOTCR1
FD71 BCKSPC
FD75 NXTCHAR
FD92 PRA1
FDB3 XAM
FDD1 ADD
FDED COUT
FE04 BLANK
FE1D SETMDZ
FE36 VFY
FE75 A1PC
FE84 SETNORM
FE8D INPRT
FE9B IOPRT
FEB3 BASCONT
FEC4 STEPZ
FEED WRBYTE
FF0A RD2
FF3F RESTORE
FF59 OLDRST
FF7A CHRSRCH
FFA2 NXTBS2
FFC7 ZMODE
FD7E CAPTST
FD96 PRYX2
FDB6 DATAOUT
FDDA PRBYTE
FDF0 COUT1
FE0B STOR
FE20 LT
FE58 VFYOK
FE78 A1PCLP
FE86 SETIFLG
FE93 SETVID
FEA7 IOPRT1
FEB6 GO
FECA USR
FEEF WRBYT2
FF16 RD3
FF44 RESTR1
FF65 MON
FF8A DIG
FFA7 GETNUM
FFCC CHRTBL
FD84 ADDINP
FDA3 XAM8
FDC5 RTS4C
FDE3 PRHEX
FDF6 COUTZ
FE17 RTS5
FE22 LT2
FE5E LIST
FE7F A1PCRTS
FE89 SETKBD
FE95 OUTPORT
FEA9 IOPRT2
FEBF REGZ
FECD WRITE
FEF6 CRMON
FF2D PRERR
FF4A SAVE
FF69 MONZ
FF90 NXTBIT
FFAD NXTCHR
FFE3 SUBTBL
FD8E CROUT
FDAD MOD8CHK
FDC6 XAMPM
FDE5 PRHEXZ
FE00 BL1
FE18 SETMODE
FE2C MOVE
FE63 LIST2
FE80 SETINV
FE8B INPORT
FE97 OUTPRT
FEB0 XBASIC
FEC2 TRACE
FED4 WR1
FEFD READ
FF3A BELL
```

FF4C SAV1
FF73 NXTITM
FF98 NXTBAS
FFBE TOSUB

SYMBOL TABLE (ALPHABETICAL ORDER)

003D A1H
FE7F A1PCRTS
0040 A3L
0044 A5L
FBF4 ADVANCE
002A BAS2L
0029 BASH
FD71 BCKSPC
FE00 BL1
FC10 BS
F9BA CHAR2
0024 CH
C059 CLRANO
FC9C CLREOL
F83C CLRSC3
FDED COUT
FC62 CR
0025 CV
F8A5 ERR
FB97 ESCOLD
F9A6 FMT2
0026 GBASL
FD6A GETLN
FCC9 HEADR
F819 HLINE
0200 IN
F882 INSDS1
C000 IOADR
03FE IRQLOC
C000 KBD
0038 KSWL
0400 LINE1
0000 LOC0
FE22 LT2
C053 MIXSET
F8C2 MNNDX2
FF69 MONZ
FA81 NEWMON
FD5F NOTCR1
FF98 NXTBAS
FD75 NXTCHAR
FA59 OLDPRK
FE97 OUTPRT
F956 PCADJ3
0095 PICK
F910 PRADR1
F930 PRADR5
FDDA PRBYTE
FDE3 PRHEX
F8DB PRNTBL
0033 PROMPT
03F4 PWREDUP
FF16 RD3
FD35 RDCHAR
FAD7 REGDSP
FF3F RESTORE
004F RNDH
F87F RTMSKZ
F961 RTS2
003C A1L
003F A2H
0043 A4H
0045 ACC
03F5 AMPERV
FBC1 BASCALC
E000 BASIC
FBD9 BELL1
FE04 BLANK
FD62 CANCEL
002E CHKSUM
FCA0 CLEOL2
C05B CLRAN1
FC42 CLREOP
F832 CLRSCR
FDF0 COUT1
FEF6 CRMON
FDB6 DATAOUT
FC2C ESC1
FD2F ESC
002E FORMAT
F856 GBCALC
FFA7 GETNUM
C057 HIRES
FC58 HOME
FB2F INIT
F88C INSDS2
FEA7 IOPRT1
FA40 IRQ
FD1B KEYIN
002F LASTIN
FE5E LIST
0001 LOC1
FE20 LT
F9C0 MNEML
F8C9 MNNDX3
FF65 MON
03FB NMI
FB94 NOWAIT
FF90 NXTBIT
FFAD NXTCHR
FF59 OLDRST
C064 PADDL0
F95C PCADJ4
F80E PLOT1
F914 PRADR2
F94A PRDL2
FB1E PREAD
FDE5 PRHEXZ
F8D4 PRNTOP
FD96 PRYX2
FAA6 PWRUP
FCFD RDBIT
FD0C RDKEY
FEBF REGZ
FF44 RESTR1
004E RNDL
F831 RTS1
FBFC RTS3
FE78 A1PCLP
003E A2L
0042 A4L
FD84 ADDINP
FB60 APPLEII
FBD0 BASCLC2
E003 BASIC2
FBE4 BELL2
FA4C BREAK
FD7E CAPTST
FF7A CHRSRCH

FC9E CLEOLZ
C05D CLRAN2
CFFF CLRROM
F836 CLRTOP
FDF6 COUTZ
0037 CSWH
FF8A DIG
FBA5 ESCNEW
FA9B FIXSEV
F847 GBASCALC
F8A9 GETFMT
FEB6 GO
C055 HISCR
F89B IEVEN
FE8B INPORT
F8D0 INSTDSP
FEA9 IOPRT2
C010 KBDSTRB
FD21 KEYIN2
002F LENGTH
FE63 LIST2
C056 LORES
002E MASK
FA00 MNEMR
FDAD MOD8CHK
FE2C MOVE
FAA3 NOFIX
FCBA NXTA1
FFA2 NXTBS2
F8F5 NXTCOL
FCE2 ONEDLY
F954 PCADJ2
003B PCH
F800 PLOT
F926 PRADR3
F94C PRBL3
FB25 PREAD2
F8F9 PRMN2
F944 PRNTX
C070 PTRIG
FCFA RD2BIT
FCEE RDBYT2
FAE4 RDSP1
F938 RELADR
FADA RGDSP1
FB19 RTBL
FBEF RTS2B
FCC8 RTS4B
FE75 A1PC
0041 A3H
0045 A5H
FDD1 ADD
002B BAS2H
FEB3 BASCONT
0028 BASL
FF3A BELL
03F0 BRKV
F9B4 CHAR1
FFCC CHRTBL
FC46 CLEOP1
C05F CLRAN3
F838 CLRSC2
0030 COLOR
FD8E CROUT
0036 CSWL
FB02 DISKID
FB9B ESCNOW
F962 FMT1
0027 GBASH
FD67 GETLNZ
002C H2
F81C HLINE1
FA6F INITAN
FE8D INPRT
0032 INVFLG
FE9B IOPRT
FB88 KBDWAIT
0039 KSWH
FC66 LF
002C LMNEM
C054 LOWSCR
C052 MIXCLR
F8BE MNNDX1
0031 MODE
07F8 MSLOT
FD3D NOTCR
FCB4 NXTA4
FAC7 NXTBYT
FF73 NXTITM
FE95 OUTPORT
F953 PCADJ
005A PCL
FD92 PRA1
F92A PRADR4
F948 PRBLNK
FF2D PRERR
F941 PRNTAX
F940 PRNTYX
FAFD PWRCON
FF0A RD2
FCEC RDBYTE
FEFD READ
FA62 RESET
002D RMNEM
F80C RTMASK
FB2E RTS2D
FDC5 RTS4C
FE17 RTS5
FC2B RTS4
FC76 SCRL1
F879 SCRN2
C05C SETAN2
FE86 SETIFLG
FE18 SETMODE
FB6F SETPWRC
002F SIGN
0049 SPNT
FE0B STOR
C060 TAPEIN
FEC2 TRACE
FECA USR
FE58 VFYOK
F828 VLINE
FCA8 WAIT
0022 WNDTOP
FEEF WRBYT2
FDA3 XAM8
FB11 XLTBL
0034 YSAV
FC8C SCRL2
FC70 SCROLL
C05E SETAN3
FE80 SETINV
FE84 SETNORM
FB39 SETTXT
FABA SLOOP
0048 STATUS
FBF0 STORADV
C020 TAPEOUT
C050 TXTCLR
03F8 USRADR
FBFD VIDOUT
FC24 VTABZ
FCAA WAIT3
0021 WNDWDTH
FEED WRBYTE
FDC6 XAMPM
0046 XREG
FCDB ZERDLY
FF4C SAV1
FC95 SCRL3
C058 SETANO
F864 SETCOL
FE89 SETKBD
FAA9 SETPG3
FE93 SETVID
03F2 SOFTEV
FEC4 STEPZ
FFE3 SUBTBL
FB09 TITLE
C051 TXTSET
002D V2
FB78 VIDWAIT
FC22 VTAB
0023 WNDBTM
FED4 WR1

```
FECD WRITE
FDB3 XAM
0047 YREG
FFC7 ZMODE
FF4A SAVE
F871 SCRN
C05A SETAN1
FB40 SETGR
FE1D SETMDZ
FAAB SETPLP
FB4B SETWND
C030 SPKR
FB65 STITLE
FB5B TABV
FFBE TOSUB
FC1A UP
FE36 VFY
F826 VLINEZ
FCA9 WAIT2
0020 WNDLFT
FCD6 WRBIT
FCE5 WRTAPE
FEB0 XBASIC
0035 YSAV1

SYMBOL TABLE SIZE
2589   BYTES USED
2531   BYTES REMAINING

SLIST 4A
```

H. K. $ 26.00

APPLE II

監督程式剖視

郭重松 編譯

波前電腦公司